REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 074-0188*

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** 18 Feb 1998	**3. REPORT TYPE AND DATES COVERED** Final Report 18 Sep 1996 - 18 Sep 1997

4. TITLE AND SUBTITLE Extraction of garment manufacturing data from 3D whole body scans	**5. FUNDING NUMBERS**
6. AUTHOR(S) Michael L. McLean Sr. Benjamin Newsom	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Haas Tailoring Company 3425 Sinclair Lane Baltimore, MD 21213	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Logistics Agency MMPRT Room 3135 8725 John J. Kingman Rd #2533 Fort Belvoir, VA 22060-6221	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES	19980623 124
12a. DISTRIBUTION / AVAILABILITY STATEMENT A - Approved for public release; distribution unlimited.	**12b. [illegible] E**

13. ABSTRACT *(Maximum 200 Words)*

The project is to develop a computer-based, interactive measurement and posture data acquisition system. The system will be implemented as a Microsoft-Windows(95/NT) computer program which accepts 3D whole body scan data and produces manufacturing-oriented measurement and posture specifications. For the men's uniform there are approximately 24 different measurements or posture specifications which may be required. For the women's uniform there are approximately 30 different measurements or posture specifications to handle. As measurements and posture specifiers are derived, they are "validated" against statistical norms and ranges. The data for these statistical norms on human build resides in the Haas Tailoring anthropomorphic data base of over 8,000 military personnel and over 75,000 members of the general population. These validated measurements and posture specifications will be automatically submitted to a pattern design system to produce a special measure pattern. Systems under considerations are the Haas Tailoring Expert System, the Gerber Garment Made-to-Measure System, and/or other garment CAD systems.

14. SUBJECT TERMS tailoring; made-to-measure; special measurement; gerber; clothing; 3D whole body data; measurement extraction			**15. NUMBER OF PAGES** 160 **16. PRICE CODE**
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

DTIC QUALITY INSPECTED 1

DLA-ARN SHORT-TERM PROJECT REPORT

Extraction of Garment Manufacturing Data From 3D Whole Body Scans

DDFG-T2-P5

Contract Number	SPO 10095-D-1044
Contractor	Haas Tailoring Company
Delivery Order Number	0004
Delivery Order Title	Extraction of Garment Manufacturing Data From 3D Whole Body Scans
CDRL #	A005
CDRL Title	Technical Report, Final Report
Reporting Period	September 18, 1996 – September 17, 1997
Report Date	February 18, 1998
Name Of PI	Michael McLean
e-mail	Mike.mclean@internetmci.com
Phone	410/732-3800
Fax	410/732-9310
Address	3425 Sinclair Lane, Baltimore, MD 21213

Final Technical Report

Date: February 1998

Extraction of Garment Manufacturing Data From 3D Whole Body Scans

Sponsor: Defense Logistics Agency

DDFG T2-P5

Principal Investigator: Michael McLean

Principal Researcher: Benjamin Newsom

Haas Tailoring Company

TABLE of CONTENTS

1 Executive Summary

From September 1996 to September 1997, Haas Tailoring Company conducted a Short Term Research Project for the Defense Logistic Agency's Apparel Research Network entitled *Extraction of Garment Manufacturing Data From 3D Whole Body Scans.* The focus of this research initiative was to create a software system, which automatically extracts apparel-related measurements, and body shape descriptions from a whole body scan. The fundamental mission of this effort is to provide consistent and accurate apparel-related body measurements and posture data in a timely manner to Recruit Induction Centers for the issue of military dress uniforms. Taking the approach of combining a whole body scanner with analysis software in the early stages of a recruit's induction process will:

- Reduce manufacturer cycle time, thereby reducing the delivery time;
- Improve the overall quality of manufactured items;
- Improve utilization of uniform material by reducing scrap/rework requirements; and
- Reduce the manufacturing costs of special measurement items.

The whole body scans of military subjects were performed at the U.S. Army Personnel Center in Natick, Massachusetts using a Cyberware whole body scanner. Haas Tailoring Company developed object-oriented computer software which builds three-dimensional (3D) mathematical models and provides various methods for analyzing and manipulating these mathematical models. In addition, Haas developed a hierarchy of software classes with methods and properties that define the set of measurements and posture descriptions necessary to manufacturer a military dress uniform. Methods in these classes automatically extract the requisite manufacturing data based upon a combination of a tailor's traditional view of a subject and recent anthropometric analyses perform on U. S. Army personnel. Haas Tailoring Company is continuing to expand, refine, and incorporate new measurement and posture information extraction algorithms to address the wide variety of potential military subjects.

2 Introduction

2.1 Purpose and Scope

The DDFG-T2-P5 Short Term Project conducted at Haas Tailoring Company created a manufacturer-oriented data acquisition system. This system acquires and validates the measurements and posture determinations required defining a given uniform order. The data acquisition system includes converting 3-D Scan Data from the Cyberware 3-D full body scanner into usable apparel body measurements and posture determinations that are compatible with the existing military clothing sizing systems. This system could be part of a point of sale system that resides in Recruit Induction Centers (RIC). The acquisition of accurate measurement and posture specifications is the first step of the apparel manufacturing process and is crucial to all downstream manufacturing process steps. A mistake or misunderstanding can be corrected or clarified at this stage for a minimal cost. If allowed to propagate to later manufacturing steps, the alteration and rework costs are substantial. The core of this project is a software program which utilizes a 3D whole body scanner to acquire measurement and posture data, analysis algorithms to determine manufacturing-oriented parameters, and becomes the front-end to an automated special measures design system for dress uniform jackets, pants, and skirts.

2.2 Apparel Manufacturer's Objectives

The most significant result of this project is consistently accurate measurements and posture descriptions. These form the basis of all future work to be performed by operations "downstream" in the manufacturing process. With this project we are moving quality and consistency checks into the early stages of the manufacturing process. Consistent and accurate measurement and posture data:

- Reduce manufacturer cycle time, thereby reducing the delivery time;
- Improve the overall quality of manufactured items;
- Improve utilization of uniform material by reducing scrap/rework requirements; and
- Reduce the manufacturing costs of special measurement items.

2.3 Project Objectives

2.3.1 Acquire Whole Body 3D scan data sets.

Whole body scans will be taken of "baseline" individuals as well as a random selection of military personnel.

2.3.2 Identify location of manufacturing-oriented points of measurement and posture specifiers within scan data.

Adapt existing and create new "shape fitting" software to recognize points of measurement such as "chest", "waist", and "seat", and posture specifiers such as "shoulder slope", "prominent seat", "prominent calves", and "bow-leg/knock-knee".

2.3.3 Extract garment-specific, manufacturing-oriented measurement and posture specifications.

Develop software algorithms for extracting garment-specific (coat, pant, and skirt) measurements and posture specifiers. Modify existing software to perform a measurement ratio analysis and determine a confidence index on the combination of measurements and posture specifiers.

2.3.4 Validate extracted measurement and posture specifications.

Automatic measurement and posture calculations will be verified against tape measurements of "baseline" individuals. In addition, the random candidate measurements will be statistically compared to the existing Haas Tailoring anthropomorphic database of over 8,000 military personnel and over 75,000 members of the general population.

2.3.5 Integrate automatically acquired measurement and posture specifications with pattern design system.

The measurement and posture specifications gleaned from the 3D whole body scans will be formatted and submitted to a pattern design system to automatically select and, if required, alter a pattern so that it conforms to the existing military clothing sizing system. Systems under considerations are the Haas Tailoring Expert System, the Gerber Garment Made-to-Measure System, and/or other garment CAD systems.

2.4 Project Results

The project began on September 18, 1996 as a 12-month project with an approximate allocation of one man-year of labor. The average weekly distribution of labor was as follows:

Project Management	0.5 day/week
Tailoring Expertise	1.0 day/week
Software Engineering	4.0 days/week

The software being created was named AMS as an acronym for Apparel Measurement System. As of the end of June 1997, the AMS software could build a 3D mathematical model of a whole body scan taken by a Cyberware Whole Body Scanner and stored in the Stanford PLY format. From this 3D model, AMS could automatically extract a limited set of measurements and body shape information necessary for apparel design and manufacturer.

As of September 17, 1997, the end of year one, the AMS system could automatically extract all of the measurements specified in the "Sample Measurement/Posture Description File" section of this report. This measurement/posture description file contains the information necessary to be submitted to an apparel pattern design system and is formatted in the de facto standard of a Microsoft Windows ".ini" file. An ini file is easily parsed by any Window-based software package. It should be noted that all of the apparel pattern design systems under consideration run under/on the Microsoft Windows operating system.

3 Operating Environment

The minimum configuration required for AMS is:

- Windows 95 on a 100Mhz Pentium, 32MB RAM, 2MB Video Card, 800x600 Monitor and access to the scan data.

Access to the scan data can be achieved via a CD-ROM drive, networked machine, or local hard disk with enough capacity to hold the scans [Note: the operating system, etc. will probably take up 400MB of disk space]). At current market prices, the minimum configuration costs less than $1,000.

At the current stage of implementation, the interactive version of AMS requires that a computer operator must be familiar enough with Windows 95 to select and run the application. From within the application, the operator must understand how to select a menu option of opening a file (the PLY representation of a scan).

The background or batch version of AMS requires no user interaction and can be executed by any other Windows-based program with the scan file name being given as a command-line argument.

4 Measurement/Manufacturing Data Extraction Approach

4.1 3D Model

The AMS software creates a three dimensional mathematical model from the points and faces identified in the Stanford PLY file representation of a Cyberware scanner's scanned "cloud of points". The model is an ordered, connected, but not necessarily closed set of points, planes, faces and edges. The model includes methods for taking "slices" through the model from any and all perspectives. These slices can be further analyzed, converted into connected hulls, converted to convex hulls, and used as bounding slices for additional, regional analysis of the scan.

4.2 Extraction Algorithms

There are certain fundamental regions of the body such as neck, shoulders, chest, wrists, waist, seat, thighs, knees and ankles that make up an overall "fingerprint" (anthropomorphic specification set) for each individual. These regions and their inter-relationships determine base measurements, measurement ratios, and posture indicators which, when combined, specify the size and shape of special-measure patterns. Software algorithms were developed to identify the specific "points of measurement" within each of these fundamental body regions.

The intelligent set of three-dimensional points, planes, faces and edges in our 3D model provide the fundamental tools necessary to extract the areas of interest to a special measure pattern designer and manufacturing operation. These areas of interest are the basis for determining posture specifiers such as a prominent seat, a sloping shoulder, an erect versus natural spine, and normal versus knock-knee versus bowed legs.

In addition to basic size information (chest, waist, seat, sleeve length, inseam, outseam, etc.) the points of interest determine the transition shape or "drop" ratio between the regions of the body. These drop ratios are crucial in determining the appropriate fit within the existing military clothing sizing system. If alterations are required, these ratios are the basis for transforming a traditional military uniform pattern into a well fitting, "no adjustment necessary" garment. The software algorithms developed perform this ratio analysis to mimic the methods used by traditional pattern designers to create the original patterns. In addition, this analysis compares the results to known "rules of thumb" to determine the severity of deviation from standard patterns.

4.3 Extraction Algorithm Validation and Verification Approach

Currently, all of the scan files (in PLY format) from the Natick scanning sessions are analyzed by automatically extracting a subset of measurements. We store these extracted measurements in a Microsoft ".ini" file format for each scanned subject. We compare the extracted values with prior measured values for each subject to determine the accuracy. With each "run", we are refining our extraction algorithms to more closely match the measured values.

4.4 Software Development Approach

The AMS software was written using standard object-oriented software development techniques and utilized all aspects of the C++ language and standard C++ template libraries. The 3D model as well as the human and garment measurement classes are built upon a hierarchical class structure described in a later section of this report. The primary development environment was Microsoft Visual C++ V5.0 within the Visual Studio. The software development objectives were as follows:

- Automatically extract from a 3D whole body scan the measurements and manufacturing information required by an apparel manufacturer to specify and create a man's dress uniform.
- Develop the application for commercial/production deployment on readily available, low-cost computing systems.
- Develop the application in a manner that requires little training or knowledge of human anatomy or tailoring by the computer operator.
- Develop the application in a manner that it can easily integrate with other applications and computing environments in place at RICs, DPSC, and apparel manufacturers.
- Where applicable, use the state-of-the-art in software development techniques and technology such as object-oriented modeling and implementation.
- Develop the application using commercially available and supported software development tools (language, compilers, editors, GUI developers, etc.).

4.5 Development Environment

Operating Systems:

Microsoft Windows 95

Microsoft Windows NT

Sun Microsystems – SunOS

Languages/Tools:

C++ (Microsoft's Visual Studio, and GNU C++)

C++ Standard Template Library

Microsoft Access

Microsoft Visual Basic

Visual Slick Edit

Borland Brief

TGS OpenInventor

Microsoft OpenGL

Microsoft Windows API

Yourdin OO modeling methods

Booch OO analysis methods

Hardware:

a) 200 Mhz Pentium Pro, 128MB RAM, 4MB 3D Video Card, 1280x1024 monitor, 3GB HD, Ethernet LAN, 20X CD-ROM reader, ZIP drive.

b) 166 Mhz Pentium, 32MB RAM, 4MB Video Card, 1024x768 monitor, 2GB HD, Ethernet LAN, 12X CD-ROM reader, ZIP drive.

c) 100 Mhz Pentium, 64MB RAM, 4MB 3D Video Card, 1280x1024 monitor, 5GB HDs, Ethernet LAN, 16X CD-ROM reader, 4X CD-ROM writer, 1GB JAZ Drive, ZIP drive.

d) 50 Mhz-486, 8MB RAM, 2MB Video Card, 1024x768 monitor, 1GB HDs, Ethernet LAN, 2X CD-ROM reader, 90MB Iomega Bernoulli drive

e) Sun Microsystems SPARC 1+, 8MB RAM, 1152x864 monitor, 600MB HDs, Ethernet LAN, 150MB tape drive

Networking Tools/Utilities:

Local Area Network

TCP/IP, SMB, IPX, NetBUI protocols

NFS, FTP, RFTP file transfer/sharing tools

Internet

FTP, HTTP, POP3, SMTP, TCP/IP

4.6 Scan Data Exchange

Scan file exchange from Natick was accomplished via the use of 150MB quarter-inch cartridge (QIC) tapes formatted on an SGI Iris at Natick. The SGI stores data on tape in a reverse byte order from everyone else so, we had to read each tape onto a hard disk on our Sun workstation as a tar image swapping each byte-pair. Once on disk, we un-tar'd the tape image across our local area network onto a 1GB hard disk partition on the primary Windows/Intel-based development machine. Eventually, we wrote these files to a ISO 9660 standard CD-ROM.

4.7 Software Architecture

The AMS software was designed around five major components: 1) user interface; 2) 3D model data structure library (C++ classes and methods); 3) data extraction and analysis library (C++ classes and methods); 4) measurement and shape set validation library (C++ classes and methods); and 5) underlying relational database. Two versions of AMS were produced. The interactive version utilizes a traditional graphics user interface (GUI). The background or batch version of AMS uses no Microsoft specific libraries, routines or dependencies. It can be called (executed, spawned, forked, etc.) by any other program with the name of the scan file as a command line or calling argument.

The user interface and the database are unique and specific to the Microsoft Windows (95/NT) operating system and could not be easily ported to another operating system. The other three components (libraries) are written using ANSI standard C++ with no Microsoft-specific extensions (API calls). The fundamental C++ class libraries for types and data structures are from the ANSI Standard Template Library. These libraries could be easily ported to other operating systems (specifically, Unix derivatives such as Solaris or IRIX) that support the ANSI standard C++ language and template library.

The three libraries are currently implemented as static libraries "compiled into" the one executable. In future releases, it would be more efficient from a long-term development and maintenance perspective to make these into individual dlls (dynamic linked libraries). Such dlls could then be utilized (callable) from programs other than AMS.

4.8 User Interface Approach

The user interface to AMS relies heavily upon and makes extensive use of the common operating systems controls provided by Windows 95/NT. It is tightly integrated into the windowing system and the Microsoft Foundation Class (MFC) structure provided by Visual C++. This allows for fairly rapid UI development, but restricts the software's use to the Windows operating system.

The fundamental approach is based upon the "wizard" control and tools that come with MFC. On a given wizard pane, the user selects the task they want to perform from a set of radio buttons then selects the "next" button that sends them to the next logical screen based upon their selection.

Because we are focusing on creating a background measurement extraction engine, the user interface is not that important. Instead, it only has to be rudimentary and functional enough to test our algorithms in a research context. Once the algorithms in the model, extraction and verification libraries are shown to be correct (or correct enough), a different user interface (control program) wrapper needs to be created (i.e., move from a prototype to a production UI).

When the transition occurs to move from a prototype UI to a production UI, we strongly recommend that either Java or Visual Basic be used as the implementing language. Java allows for portability across platforms, but lacks functionality in comparison to Visual Basic. Visual Basic is quite robust in its ability to interface and interact with almost any application that runs on the Windows 95/98/NT operating system.

5 Measurement/Manufacturing Data Extraction Capabilities

The following sections describe the capabilities as of the end of the first year of research of the apparel measuring system (AMS) software created by Haas Tailoring

Company under its Apparel Research Network's DDFG, MPG2, T2P5 project titled "Extraction of Garment Manufacturing Data From 3D Whole Body Scans."

The data source for automatic measurements is 3D whole body scanned data stored in PLY format files. (Note: PLY format is a graphical data storage format defined by a graduate student at Stanford and used by Cyberware scan post processing software.) Although we can determine the geometric orientation of the individual, we assume the pose is the "relaxed" pose facing in a specific direction (as was done with all of the Natick scans). We also assume the scan is of a "normal" (i.e., non-deformed, non-tramatized) individual. At this point, most of the circumference-type measurements can handle the presence or absence of limbs, but the current software is not robust enough to handle all injury or deformities that may occur to military personnel.

We automatically extract measurements based upon the garment being ordered. An example of why this is important is the waist measurement. The waist measurement for a coat may be larger than the waist measurement for pants. Or, they may be the same size. Either situation is acceptable and downstream pattern creation rules handle the delta between the measurements. Also, by focusing on the garment being sought (ordered) we can limit the number of measurements to be acquired to the minimal set needed.

The program structure and the underlying data structure are setup and designed to handle the following garments:

Men's Coat
Men's Pants
Men's Shirt

Women's Coat
Women's Slacks
Women's Skirt

Women's Blouse (shirt)

5.1 Current General Measurements

Chest Circumference

Chest Height

Crotch Height

Height

Neck Circumference

Neck Height

Overarm Circumference

Point To Point

Point To Point Height

Seat Circumference

Seat Height

Shoulder Circumference

Shoulder Height

Waist Circumference

Waist Height

Waist Height Back

Waist Height Front

5.2 Current Coat Measurements

Full Length

Chest

Neck

Overarm

Point To Point

Seat

Shoulder Angle Description

Shoulder (left)

Shoulder (right)

Waist = 31.513

5.3 Current Pant Measurements

Inseam (left)

Inseam (right)

Outseam (left)

Outseam (right)

Rise

Seat

Waist

Waist Height (back)

Waist Height (front)

5.4 3D Model Classes and Methods

5.4.1 3D Geometry Classes

Point3D	base3d.h
Vector3D	base3d.h
Cylinder3D	ams3d.h
LineRep3D	ams3d.h
LineSegment3D	ams3d.h
Plane3D	ams3d.h
PlanePts	ams3d.h
Point2D	ams3d.h
MergeFind	mfalgo.h

5.4.2 Model Classes

AMSTypes	amsmodel.h
TriFace	amsmodel.h
TriFaceVector	amsmodel.h
Edge	amsmodel.h
Faces	amsmodel.h

Vertices	amsmodel.h
EdgeVector	amsmodel.h
AMSModel	amsmodel.h
Bounds	amsmodel.h

5.4.3 Model Rendering Classes

AMSModelRenderer amsrend.h

Used to render (draw) a generic bitmap representation of a specific view of the model's vertices.

5.4.4 Base I/O Class

AMSModelIOamsio.h

This is the base class for reading in and writing out coordinate and 3D geometry information to files. It is based upon the C++ I/O streams classes. The format-specific file I/O classes are derived from this class.

5.4.5 PLY I/O Stream Classes

PLYModelIO plyio.h

The PLY I/O Stream class is used to read in a PLY file storing the vertices and faces as specified in the file into the 3D model structures. As part of reading the file we calculate and store a variety of bounding information as well as identify all of the edges present in the coordinate information.

mungeistream munge.h

Note: Because the SGI has a "Big Indian" addressing architecture and the Intel world has a "Little Indian" addressing architecture, the binary format of the PLY files had to be swapped. Specifically, the byte order of a float is reversed. We dynamically swap bytes as we read in a binary PLY file produced by the Cyberware software.

5.5 Extraction Classes and Methods

The extraction methods are based upon traditional measurement taking techniques used by Haas Tailoring Company and from the 1988 U.S. Army Anthropometric Survey (ANSUR) conducted by Anthropology Research Project, Inc. of Ohio. In addition, we have included some shape assessment algorithms which derive from prior work on the design and manufacture of nautical sails for racing sailboats.

The extraction code consists of two C++ class hierarchies and their interrelationships. One class hierarchy is based upon the types of human anatomical measurements and body shape determinations a tailor might make. Deriving from base (generic) human classes, the top of this hierarchy is a male measurements class and a female measurements class. Given an AMSModel, the methods for these classes will automatically extract body measurements. The AMSModel methods provide a first level of information extraction tools. The body measurement methods are a second level of extraction for "straight forward" measurements.

The other class hierarchy is based upon the types of measurements and analyses that are required by a specific garment. A given garment only needs a subset of possible male/female measurements; therefore, an efficient approach to extracting data is to only extract what is needed for the garment being procured. In addition, these methods provide a third level of analysis where a particular garment measurement or shape determination needs to combine a number of body measurements to determine one value.

MaleMeasurements and FemaleMeasurements are derived from HumanMeasurements which is derived from the base Measurement. The ArmMeasurements and the LegMeasurements are also derived from the base Measurement. The HumanMeasurements class contains variables leftArm and rightArm which are instances of ArmMeasurements classes. Similarly, the HumanMeasurements class contains variables leftLeg and rightLeg which are instances of LegMeasurements classes. This represents a combination of both inheritance and "contains" type of object relationships.

5.5.1 Measurement Classes

MaleMeasurements	malemeasurements.h
FemaleMeasurements	femalemeaurements.h
HumanMeasurements	humanmeasurements.h
ArmMeasurements	armmeasurements.h
LegMeasurements	legmeasurements.h
Measurement	measurement.h

5.5.2 Garment Classes

CoatMeasurements	coatmeasurements.h
PantMeasurements	pantmeasurements.h
ShirtMeasurements	shirtmeasurements.h
SlackMeasurements	slackmeasurements.h
SkirtMeasurements	skirtmeasurements.h

5.6 Measurement/Shape Validation Classes and Methods

The measurement/shape validation code is currently being written, but it is based upon the above mentioned garment classes. The garment being procured defines the combination of measurements and shape determination that is required to specify an appropriate pattern. Based upon historical data on measurement combinations, rules of thumb used in the apparel industry, and on the 1988 ANSUR survey we can determine the likelihood of occurrence of a given combination.

5.7 Database Approach

The underlying database is not necessarily crucial at this stage of development, but will (should) become very important as the measuring system moves from being a prototype to a production system. We selected Access as the database engine for many reasons: it is prevalent among ARN partners, the AIMS software is based on Access, Access can be utilized by numerous Windows-based tools and languages, and it had a Data Access Object (DAO) framework already established in MFC.

6 Glossary

Bow-legged – when the distance between the knees in a normal stance is greater than five fingers.

Erect posture – an erect vs. natural spine (the “Z” delta from the neck point vs. the lower shoulder blades),

Knocked-knees – when the distance between the knees in a heels-together stance is greater than three fingers.

Prominent seat -- The delta between a “back-to-floor” and “front-to-floor measurement in conjunction with the delta between a “back rise” and “front rise” calculation.

Sloping shoulder -- The delta between the neck to shoulder drop of each shoulder.

7 Results With Trial Subjects

During the research and investigation period covered by this project, there were eight (8) useable scanned subjects. Please see the section in this report entitled Diagnostic Pictures. Of these eight scans, subjects N000001, N000002, N000004, N000005, and N000006 were male. The other three subjects, N000007, N000008, and N000009, were female. Although the extraction algorithms were intended to be gender neutral, the greatest emphasis was on the extraction of male measurements.

The following tables show the results from automatic extraction as compared to manual tape measure determination of these same measurements. Grading scale is A = +/- .5”, B = +/- .75”, C = +/- 1.0”, D > 1.0” as per T2-P5 Project Management’s grading specifications.

Height (inches)

Subject	Manual	Extracted	Difference	Grade
N000001	69.17	67.24	1.93	D
N000002	65.79	65.12	0.67	B
N000004	69.65	70.08	- 0.43	A
N000005	73.43	73.47	- 0.04	A
N000006	66.34	65.59	0.75	B
N000007	63.07	63.47	- 0.40	A
N000008	64.33	64.25	0.08	A
N000009	65.39	65.90	- 0.51	B

Chest (inches)

Subject	Manual	Extracted	Difference	Grade
N000001	35.91	36.85	- 0.94	C
N000002	39.09	39.66	- 0.57	B
N000004	38.78	38.63	0.15	A
N000005	47.83	56.03	- 8.20	F
N000006	39.33	39.17	0.16	A
N000007	36.02	36.34	- 0.32	A
N000008	36.22	37.40	- 1.18	D
N000009	32.48	32.70	- 0.22	A

Waist (inches)

Subject	Manual	Extracted	Difference	Grade
N000001	30.00	31.08	-1.02	D
N000002	31.02	31.77	-0.75	B
N000004	33.78	33.73	0.05	A
N000005	44.72	43.40	1.32	D
N000006	30.98	31.51	-0.53	B
N000007	26.22	27.71	-1.49	D
N000008	28.98	29.80	-0.82	C
N000009	27.20	27.84	-0.64	B

Seat (inches)

Subject	Manual	Extracted	Difference	Grade
N000001	37.91	38.45	-0.54	B
N000002	38.46	38.37	0.09	A
N000004	39.09	39.16	-0.07	A
N000005	45.20	45.47	-0.27	A
N000006	37.48	37.60	-0.12	A
N000007	36.81	36.76	0.05	A
N000008	38.35	38.69	-0.34	A
N000009	38.82	39.53	-0.71	B

Inseam (inches)

Subject	Manual	Extracted	Difference	Grade
N000001	30.43	28.92	1.51	D
N000002	31.38	30.12	1.26	D
N000004	31.81	31.01	0.8	B
N000005	33.11	32.51	0.6	B
N000006	31.10	30.50	0.6	B
N000007	27.83	28.08	-0.25	A
N000008	29.69	29.07	0.62	B
N000009	30.63	30.32	0.31	A

Point-to-Point (inches)

Subject	Manual	Extracted	Difference	Grade
N000001	19.00	19.26	- .26	A
N000002	21.50	21.13	.37	A
N000004	20.50	19.51	0.99	C
N000005	22.00	22.34	- .34	A
N000006	21.50	21.07	.43	A
N000007	16.75	17.97	- 1.22	D
N000008	17.00	18.125	- 1.13	D
N000009	16.75	17.08	- .33	A

8 Measurement/Posture Description Files

```
[Subject]
Name = n000001.sprd.rgb.edt.ply.measurements.Inches.txt

[General Measurements]
Units = Inches
Acromial Height = 55.140
Chest Circumference = 36.849
Chest Height = 48.416
Crotch Height = 28.915
Height = 67.244
Neck Circumference = 16.097
Neck Height = 57.561
Overarm Circumference = 48.208
Point To Point = 19.260
Point To Point Height = 54.140
Seat Circumference = 38.447
Seat Height = 32.422
Shoulder Circumference = 41.444
Shoulder Height = 53.390
Waist Circumference = 31.084
Waist Height = 37.993
Waist Height Back = 39.457
Waist Height Front = 37.993

[Coat Measurements]
Back Width =  0.000
Bicep =  0.000
Full Length = 28.646
Chest = 36.849
Neck = 16.097
Neck Description =
Overarm = 48.208
Point To Point = 19.260
Posture =  0.000
Posture Description =
Seat = 38.447
Seat Description =
Shoulder Angle Description = Full Sloping Shoulders
Shoulder Build Description =
Shoulder (left) =  3.421
Shoulder (right) =  3.421
Shoulder Pitch =
Sleeve Inseam (left) =  0.000
Sleeve Inseam (right) =  0.000
Sleeve Outseam (left) =  0.000
Sleeve Outseam (right) =  0.000
Waist = 31.084

[Pant Measurements]
Abdomen =  0.000
Calf (left) =  0.000
Calf (right) =  0.000
Inseam (left) = 28.915
Inseam (right) = 28.915
Knee (left) =  0.000
Knee (right) =  0.000
Outseam (left) = 37.993
Outseam (right) = 37.993
Rise =  9.078
Rise Description =
Seat = 38.447
Seat Description =
Thigh (left) =  0.000
Thigh (right) =  0.000
Waist = 31.084
Waist Height (back) = 39.457
Waist Height (front) = 37.993

=======================================================
[Subject]
Name = n000002.sprd.rgb.edt.ply.measurements.Inches.txt

[General Measurements]
Units = Inches
Acromial Height = 53.397
Chest Circumference = 39.664
Chest Height = 46.885
Crotch Height = 30.117
Height = 65.118
Neck Circumference = 15.815
Neck Height = 56.262
Overarm Circumference = 54.309
Point To Point = 21.127
Point To Point Height = 52.397
Seat Circumference = 38.369
Seat Height = 33.701
Shoulder Circumference = 45.352
Shoulder Height = 51.647
Waist Circumference = 31.773
Waist Height = 38.745
Waist Height Back = 40.148
Waist Height Front = 38.745

[Coat Measurements]
Back Width =  0.000
Bicep =  0.000
Full Length = 26.145
Chest = 39.664
Neck = 15.815
Neck Description =
Overarm = 54.309
Point To Point = 21.127
Posture =  0.000
Posture Description =
Seat = 38.369
Seat Description =
Shoulder Angle Description = Full Sloping Shoulders
Shoulder Build Description =
Shoulder (left) =  3.865
Shoulder (right) =  3.865
Shoulder Pitch =
Sleeve Inseam (left) =  0.000
Sleeve Inseam (right) =  0.000
Sleeve Outseam (left) =  0.000
Sleeve Outseam (right) =  0.000
Waist = 31.773

[Pant Measurements]
Abdomen =  0.000
Calf (left) =  0.000
Calf (right) =  0.000
Inseam (left) = 30.117
Inseam (right) = 30.117
Knee (left) =  0.000
Knee (right) =  0.000
Outseam (left) = 38.745
Outseam (right) = 38.745
Rise =  8.628
Rise Description =
Seat = 38.369
Seat Description =
Thigh (left) =  0.000
Thigh (right) =  0.000
Waist = 31.773
Waist Height (back) = 40.148
Waist Height (front) = 38.745

=======================================================
[Subject]
Name = n000004.sprd.rgb.edt.ply.measurements.Inches.txt

[General Measurements]
Units = Inches
Acromial Height = 57.465
Chest Circumference = 38.627
Chest Height = 50.457
Crotch Height = 31.010
Height = 70.079
Neck Circumference = 16.197
Neck Height = 59.847
Overarm Circumference = 57.043
Point To Point = 19.507
Point To Point Height = 56.465
Seat Circumference = 39.158
Seat Height = 35.354
Shoulder Circumference = 42.768
Shoulder Height = 55.715
Waist Circumference = 33.732
Waist Height = 43.449
Waist Height Back = 45.816
Waist Height Front = 43.449

[Coat Measurements]
Back Width =  0.000
Bicep =  0.000
```

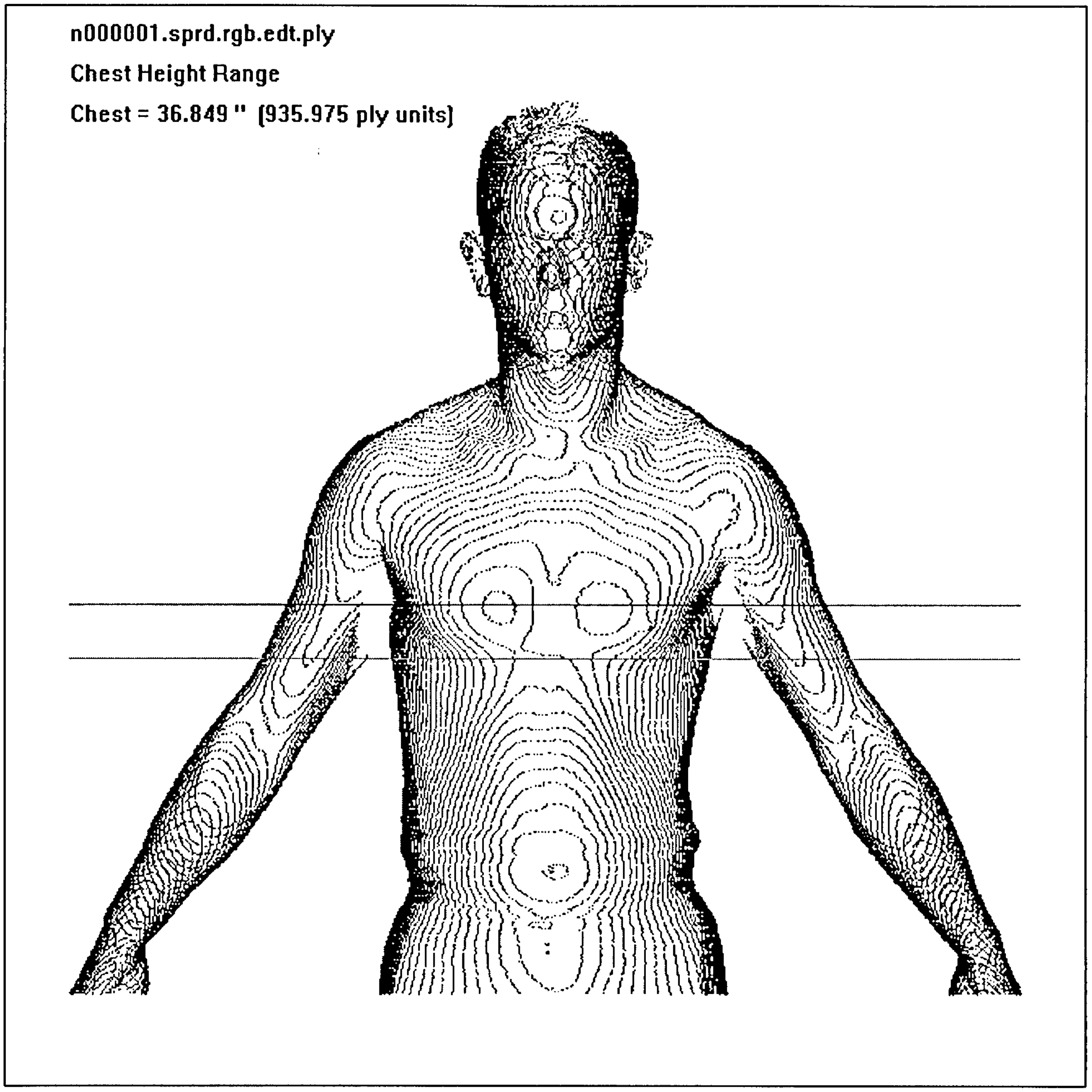
```
Full Length = 28.837
Chest = 38.627
Neck = 16.197
Neck Description =
Overarm = 57.043
Point To Point = 19.507
Posture =  0.000
Posture Description =
Seat = 39.158
Seat Description =
Shoulder Angle Description = Full Sloping Shoulders
Shoulder Build Description =
Shoulder (left) =  3.383
Shoulder (right) =  3.383
Shoulder Pitch =
Sleeve Inseam (left) =  0.000
Sleeve Inseam (right) =  0.000
Sleeve Outseam (left) =  0.000
Sleeve Outseam (right) =  0.000
Waist = 33.732

[Pant Measurements]
Abdomen =  0.000
Calf (left) =  0.000
Calf (right) =  0.000
Inseam (left) = 31.010
Inseam (right) = 31.010
Knee (left) =  0.000
Knee (right) =  0.000
Outseam (left) = 43.449
Outseam (right) = 43.449
Rise = 12.439
Rise Description =
Seat = 39.158
Seat Description =
Thigh (left) =  0.000
Thigh (right) =  0.000
Waist = 33.732
Waist Height (back) = 45.816
Waist Height (front) = 43.449

=======================================================
[Subject]
Name = n000005.sprd.rgb.edt.ply.measurements.Inches.txt

[General Measurements]
Units = Inches
Acromial Height = 60.241
Chest Circumference = 56.034
Chest Height = 51.058
Crotch Height = 32.508
Height = 73.465
Neck Circumference = 20.902
Neck Height = 63.033
Overarm Circumference = 67.973
Point To Point = 22.389
Point To Point Height = 59.241
Seat Circumference = 45.471
Seat Height = 38.819
Shoulder Circumference = 48.091
Shoulder Height = 58.491
Waist Circumference = 43.400
Waist Height = 45.548
Waist Height Back = 47.587
Waist Height Front = 45.548

[Coat Measurements]
Back Width =  0.000
Bicep =  0.000
Full Length = 30.525
Chest = 56.034
Neck = 20.902
Neck Description =
Overarm = 67.973
Point To Point = 22.389
Posture =  0.000
Posture Description =
Seat = 45.471
Seat Description =
Shoulder Angle Description = Full Sloping Shoulders
Shoulder Build Description =
Shoulder (left) =  3.792
Shoulder (right) =  3.792
Shoulder Pitch =
Sleeve Inseam (left) =  0.000
Sleeve Inseam (right) =  0.000
Sleeve Outseam (left) =  0.000
Sleeve Outseam (right) =  0.000
Waist = 43.400

[Pant Measurements]
Abdomen =  0.000
Calf (left) =  0.000
Calf (right) =  0.000
Inseam (left) = 32.508
Inseam (right) = 32.508
Knee (left) =  0.000
Knee (right) =  0.000
Outseam (left) = 45.548
Outseam (right) = 45.548
Rise = 13.040
Rise Description =
Seat = 45.471
Seat Description =
Thigh (left) =  0.000
Thigh (right) =  0.000
Waist = 43.400
Waist Height (back) = 47.587
Waist Height (front) = 45.548

=======================================================
[Subject]
```

```
Name = n000006.sprd.rgb.edt.ply.measurements.Inches.txt

[General Measurements]
Units = Inches
Acromial Height = 53.784
Chest Circumference = 39.166
Chest Height = 46.897
Crotch Height = 30.500
Height = 65.591
Neck Circumference = 17.637
Neck Height = 56.670
Overarm Circumference = 54.425
Point To Point = 21.073
Point To Point Height = 52.784
Seat Circumference = 37.602
Seat Height = 34.724
Shoulder Circumference = 45.292
Shoulder Height = 52.034
Waist Circumference = 31.513
Waist Height = 40.010
Waist Height Back = 41.944
Waist Height Front = 40.010

[Coat Measurements]
Back Width =  0.000
Bicep =  0.000
Full Length = 26.171
Chest = 39.166
Neck = 17.637
Neck Description =
Overarm = 54.425
Point To Point = 21.073
Posture =  0.000
Posture Description =
Seat = 37.602
Seat Description =
Shoulder Angle Description = Full Sloping Shoulders
Shoulder Build Description =
Shoulder (left) =  3.886
Shoulder (right) =  3.886
Shoulder Pitch =
Sleeve Inseam (left) =  0.000
Sleeve Inseam (right) =  0.000
Sleeve Outseam (left) =  0.000
Sleeve Outseam (right) =  0.000
Waist = 31.513

[Pant Measurements]
Abdomen =  0.000
Calf (left) =  0.000
Calf (right) =  0.000
Inseam (left) = 30.500
Inseam (right) = 30.500
Knee (left) =  0.000
Knee (right) =  0.000
Outseam (left) = 40.010
Outseam (right) = 40.010
Rise =  9.511
Rise Description =
Seat = 37.602
Seat Description =
Thigh (left) =  0.000
Thigh (right) =  0.000
Waist = 31.513
Waist Height (back) = 41.944
Waist Height (front) = 40.010

=======================================================
[Subject]
Name = n000007.sprd.rgb.edt.ply.measurements.Inches.txt

[General Measurements]
Units = Inches
Acromial Height = 52.041
Chest Circumference = 36.337
Chest Height = 45.377
Crotch Height = 28.083
Height = 63.465
Neck Circumference = 16.510
Neck Height = 54.326
Overarm Circumference = 61.639
Point To Point = 17.971
Point To Point Height = 51.041
Seat Circumference = 36.756
Seat Height = 31.328
Shoulder Circumference = 39.892
Shoulder Height = 50.291
Waist Circumference = 27.709
Waist Height = 39.348
Waist Height Back = 41.564
Waist Height Front = 39.348

[Coat Measurements]
Back Width =  0.000
Bicep =  0.000
Full Length = 26.243
Chest = 36.337
Neck = 16.510
Neck Description =
Overarm = 61.639
Point To Point = 17.971
Posture =  0.000
Posture Description =
Seat = 36.756
Seat Description =
Shoulder Angle Description = Full Sloping Shoulders
Shoulder Build Description =
Shoulder (left) =  3.285
Shoulder (right) =  3.285
Shoulder Pitch =
Sleeve Inseam (left) =  0.000
Sleeve Inseam (right) =  0.000
Sleeve Outseam (left) =  0.000
Sleeve Outseam (right) =  0.000
Waist = 27.709

[Pant Measurements]
Abdomen =  0.000
Calf (left) =  0.000
Calf (right) =  0.000
Inseam (left) = 28.083
Inseam (right) = 28.083
Knee (left) =  0.000
Knee (right) =  0.000
Outseam (left) = 39.348
Outseam (right) = 39.348
Rise = 11.265
Rise Description =
Seat = 36.756
Seat Description =
Thigh (left) =  0.000
Thigh (right) =  0.000
Waist = 27.709
Waist Height (back) = 41.564
Waist Height (front) = 39.348

=======================================================
[Subject]
Name = n000008.sprd.rgb.edt.ply.measurements.Inches.txt

[General Measurements]
Units = Inches
Acromial Height = 52.687
Chest Circumference = 37.397
Chest Height = 46.261
Crotch Height = 29.074
Height = 64.252
Neck Circumference = 13.381
Neck Height = 55.257
Overarm Circumference = 49.519
Point To Point = 18.125
Point To Point Height = 51.687
Seat Circumference = 38.691
Seat Height = 32.728
Shoulder Circumference = 39.860
Shoulder Height = 50.937
Waist Circumference = 29.802
Waist Height = 39.836
Waist Height Back = 41.683
Waist Height Front = 39.836

[Coat Measurements]
Back Width =  0.000
Bicep =  0.000
Full Length = 26.183
Chest = 37.397
Neck = 13.381
Neck Description =
Overarm = 49.519
Point To Point = 18.125
Posture =  0.000
Posture Description =
Seat = 38.691
Seat Description =
Shoulder Angle Description = Full Sloping Shoulders
Shoulder Build Description =
Shoulder (left) =  3.570
Shoulder (right) =  3.570
Shoulder Pitch =
Sleeve Inseam (left) =  0.000
Sleeve Inseam (right) =  0.000
Sleeve Outseam (left) =  0.000
Sleeve Outseam (right) =  0.000
Waist = 29.802

[Pant Measurements]
Abdomen =  0.000
Calf (left) =  0.000
Calf (right) =  0.000
Inseam (left) = 29.074
Inseam (right) = 29.074
Knee (left) =  0.000
Knee (right) =  0.000
Outseam (left) = 39.836
Outseam (right) = 39.836
Rise = 10.762
Rise Description =
Seat = 38.691
Seat Description =
Thigh (left) =  0.000
Thigh (right) =  0.000
Waist = 29.802
Waist Height (back) = 41.683
Waist Height (front) = 39.836

=======================================================
[Subject]
Name = n000009.sprd.rgb.edt.ply.measurements.Inches.txt

[General Measurements]
Units = Inches
Acromial Height = 54.042
Chest Circumference = 32.697
Chest Height = 47.452
Crotch Height = 30.317
Height = 65.905
Neck Circumference = 13.132
Neck Height = 56.020
Overarm Circumference = 54.381
Point To Point = 17.084
Point To Point Height = 53.042
Seat Circumference = 39.534
Seat Height = 33.947
Shoulder Circumference = 37.446
Shoulder Height = 52.292
Waist Circumference = 27.837
Waist Height = 40.861
Waist Height Back = 43.512
Waist Height Front = 40.861

[Coat Measurements]
Back Width =  0.000
Bicep =  0.000
Full Length = 25.703
Chest = 32.697
Neck = 13.132
Neck Description =
Overarm = 54.381
Point To Point = 17.084
Posture =  0.000
Posture Description =
Seat = 39.534
Seat Description =
Shoulder Angle Description = Half Sloping Shoulders
Shoulder Build Description =
Shoulder (left) =  2.977
Shoulder (right) =  2.977
Shoulder Pitch =
Sleeve Inseam (left) =  0.000
Sleeve Inseam (right) =  0.000
Sleeve Outseam (left) =  0.000
Sleeve Outseam (right) =  0.000
Waist = 27.837

[Pant Measurements]
Abdomen =  0.000
Calf (left) =  0.000
Calf (right) =  0.000
Inseam (left) = 30.317
Inseam (right) = 30.317
Knee (left) =  0.000
Knee (right) =  0.000
Outseam (left) = 40.861
Outseam (right) = 40.861
Rise = 10.545
Rise Description =
Seat = 39.534
Seat Description =
Thigh (left) =  0.000
Thigh (right) =  0.000
Waist = 27.837
Waist Height (back) = 43.512
Waist Height (front) = 40.861
```

9 Sample Diagnostic Pictures of Subjects

The following pages contain sample diagnostic pictures used to evaluate the performance of the data extraction algorithms. The pictures are for subjects:

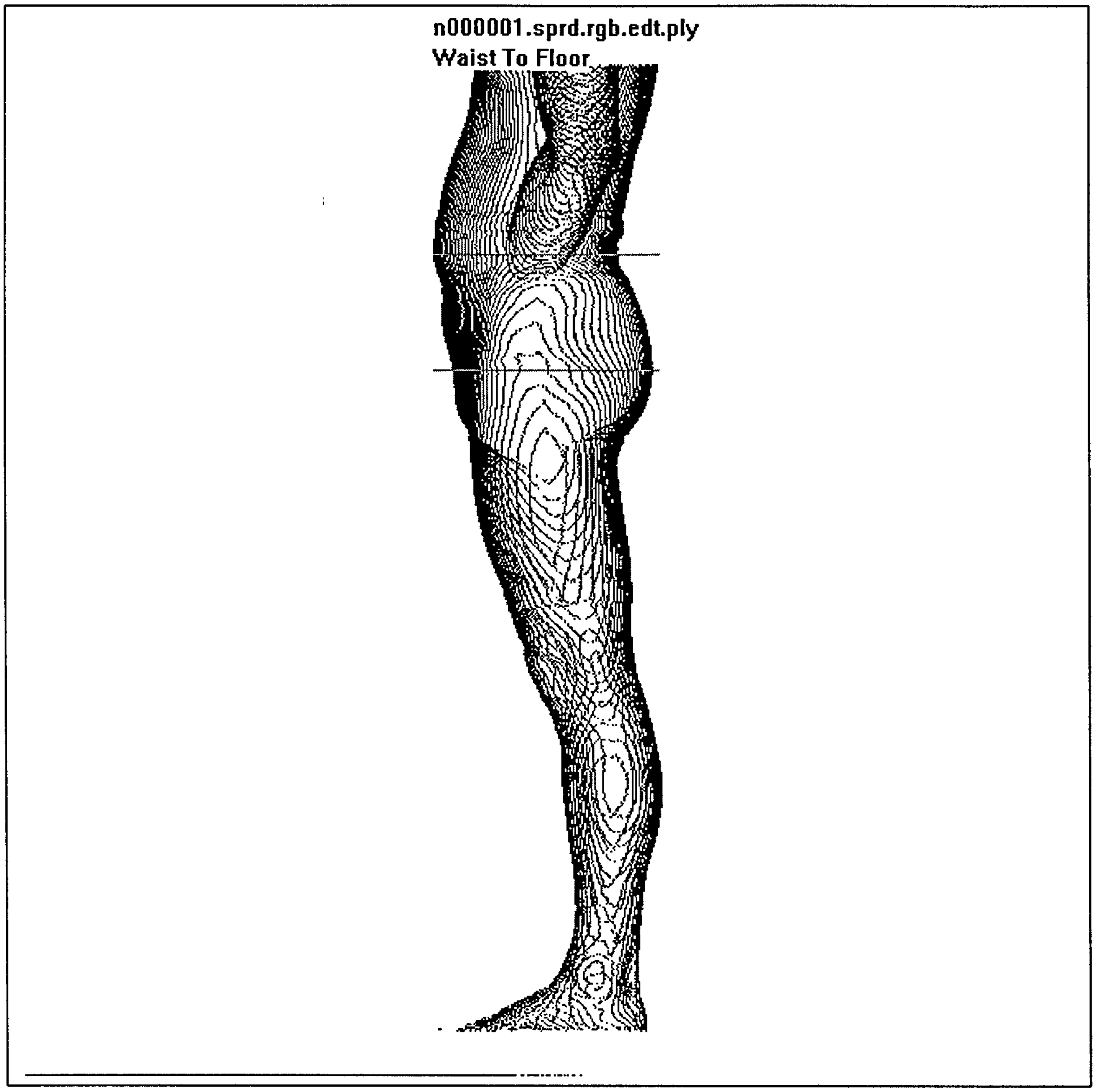
N000001
N000002
N000004
N000005
N000006
N000007
N000008
N000009

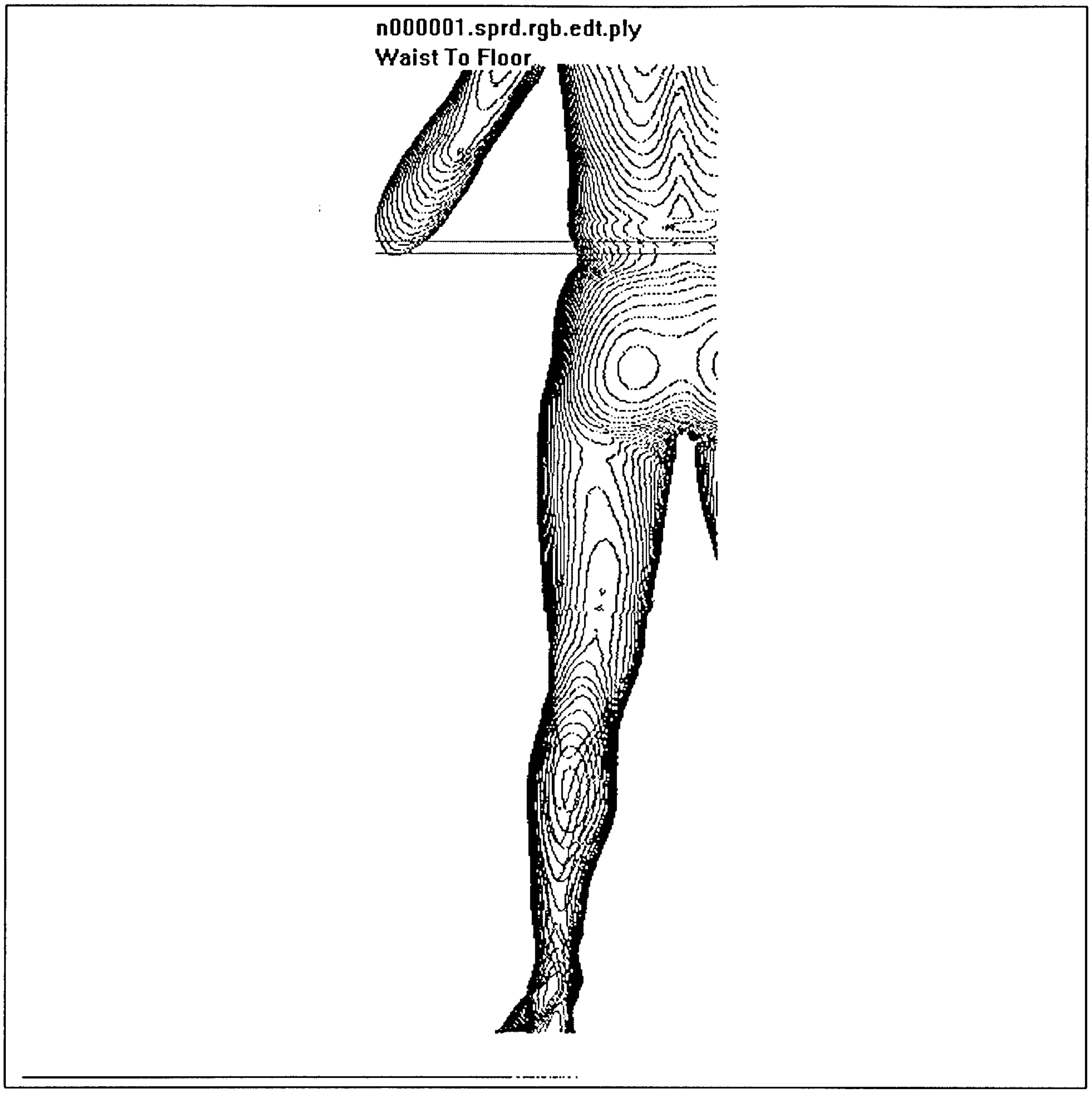


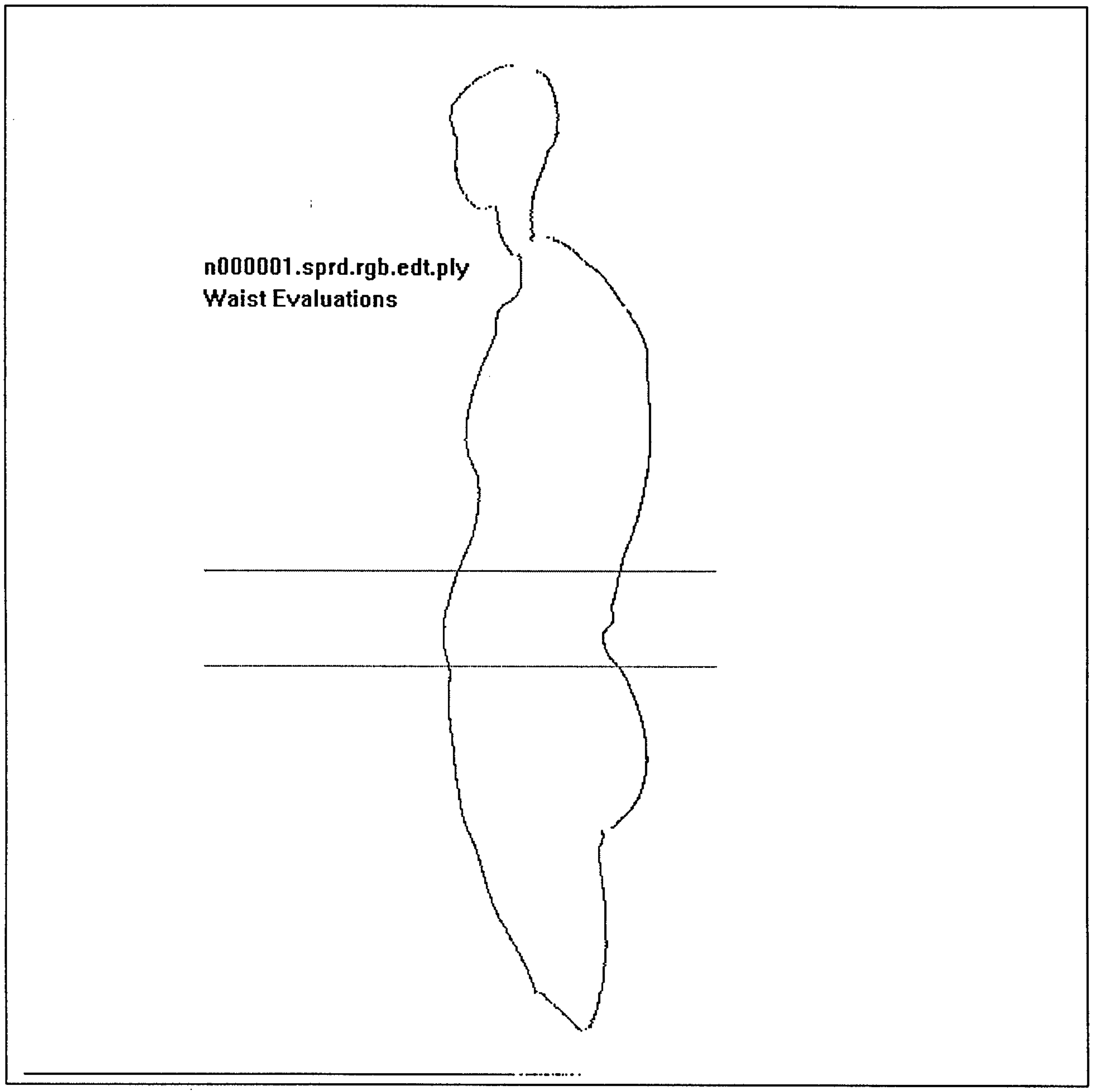
n000001.sprd.rgb.edt.ply

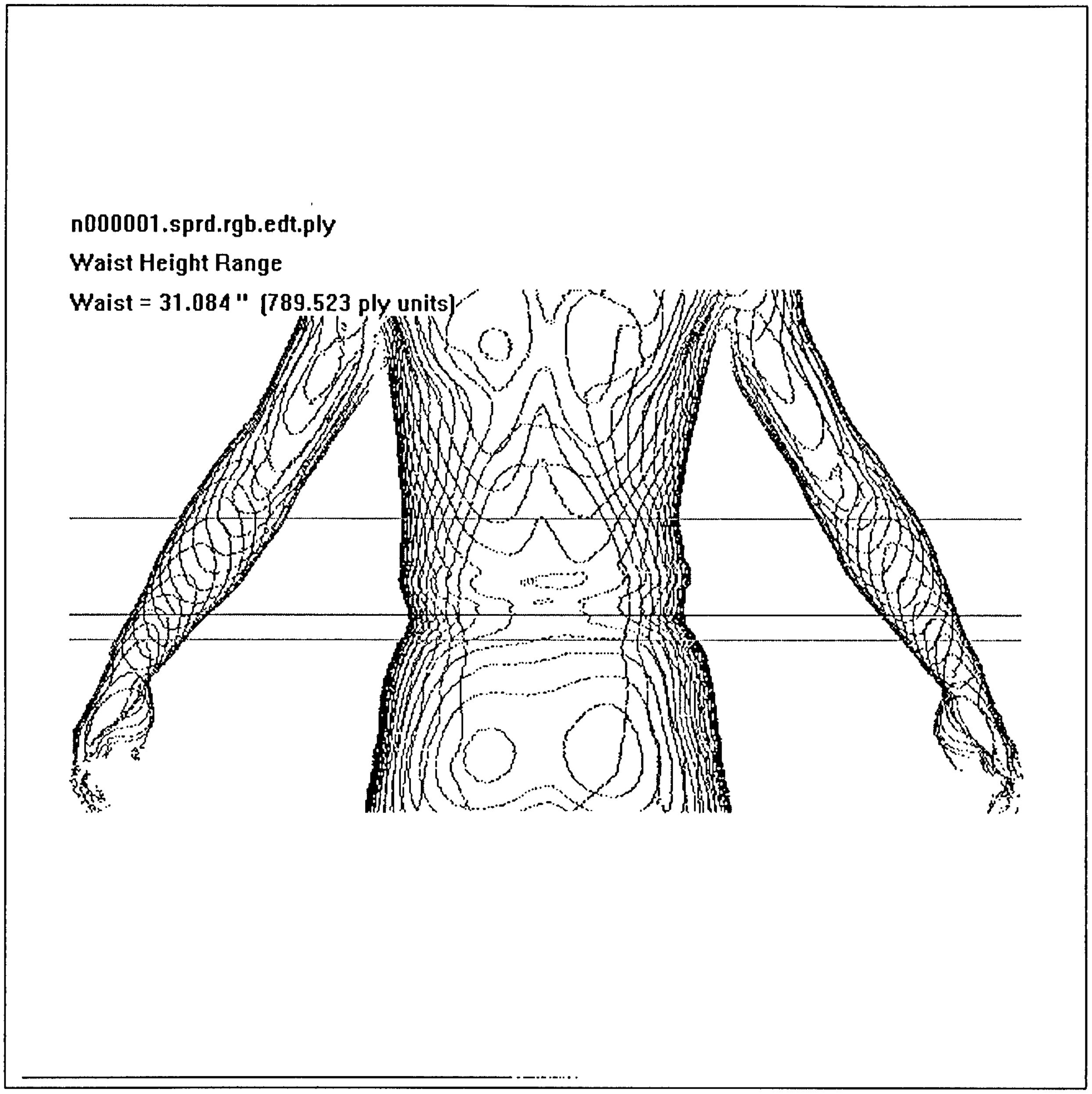
Chest Height Range

Chest = 36.849 " (935.975 ply units)


n000001.sprd.rgb.edt.ply
Waist To Floor

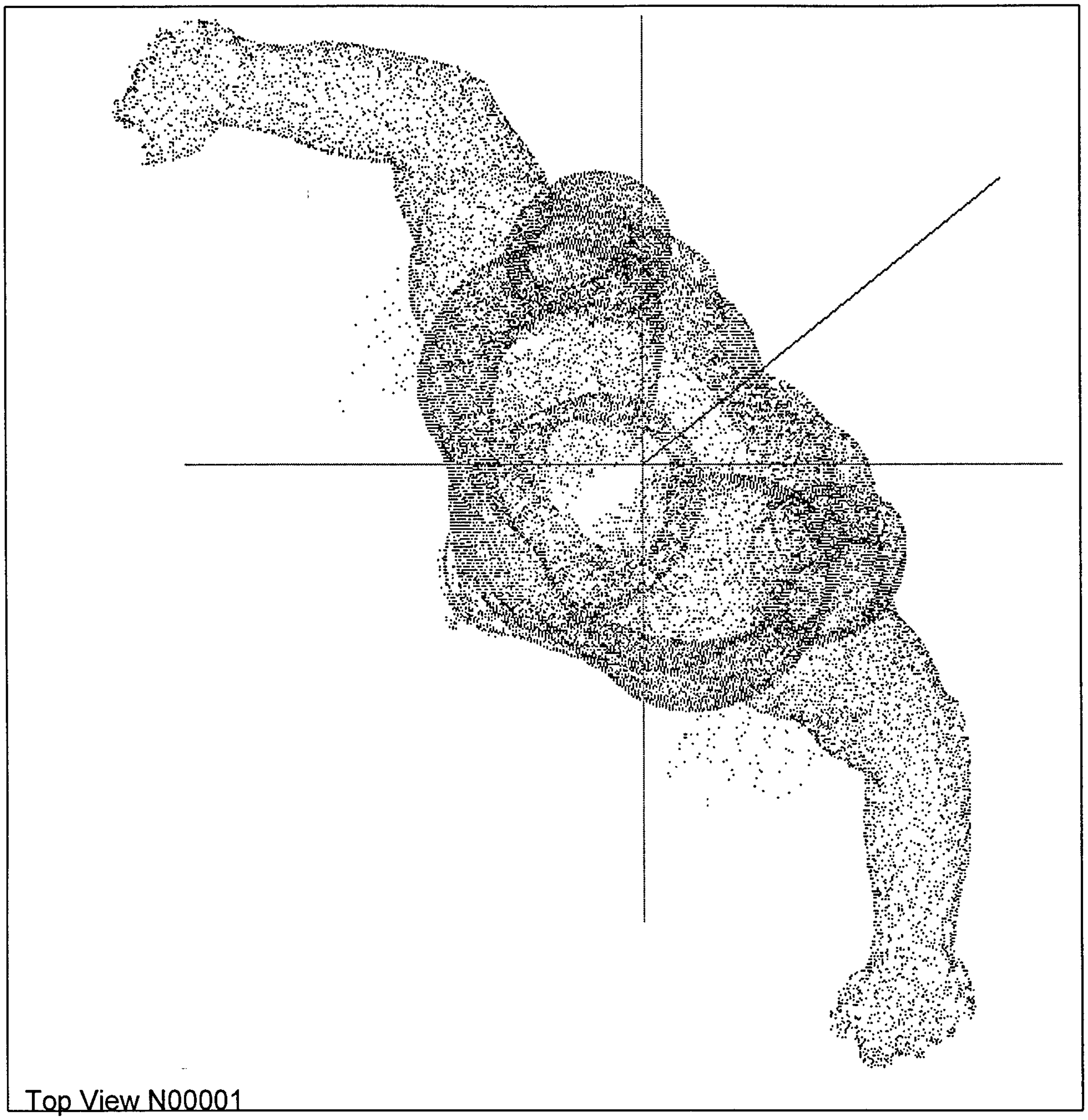

n000001.sprd.rgb.edt.ply
Waist To Floor

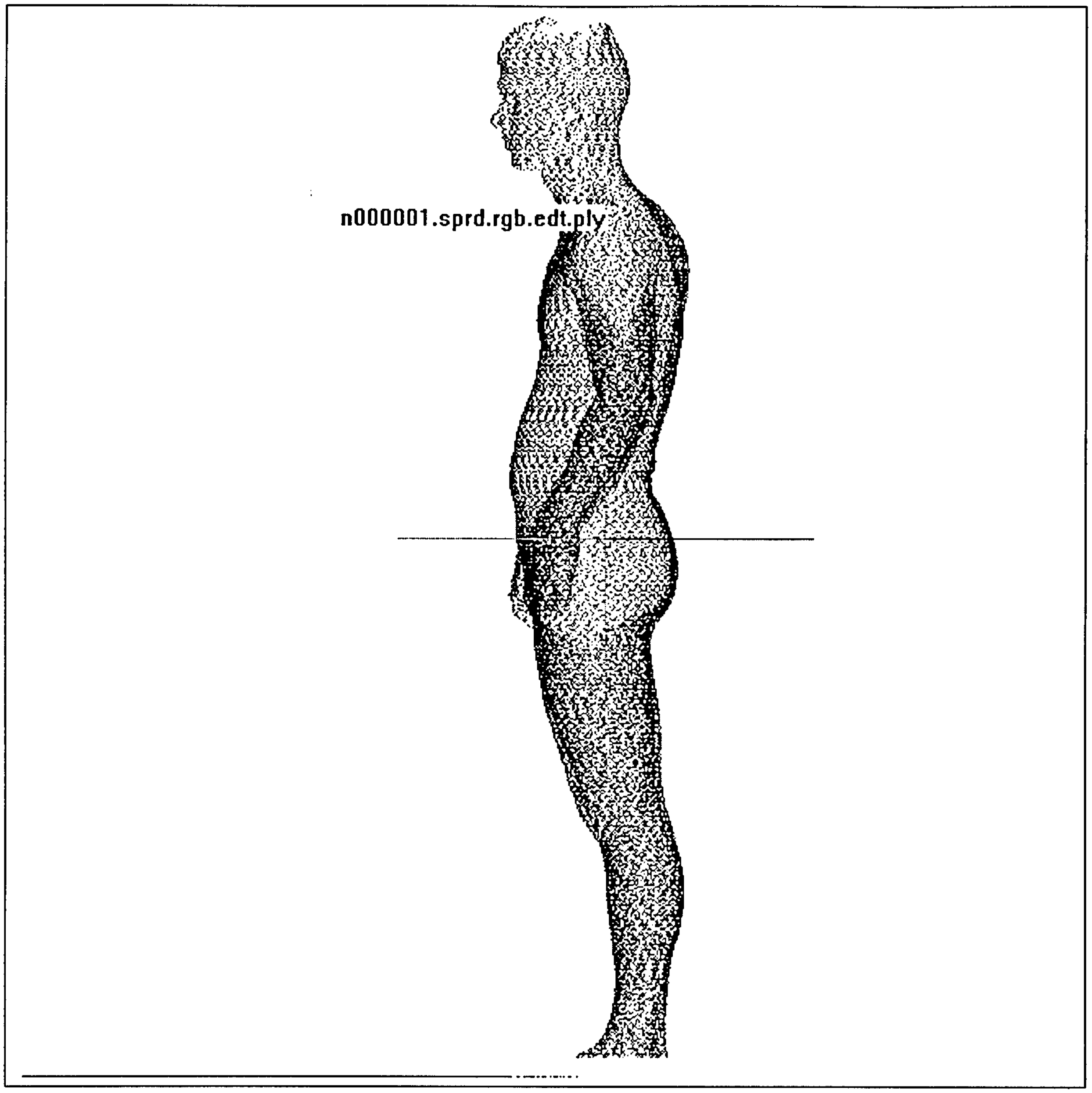

n000001.sprd.rgb.edt.ply
Waist Evaluations


n000001.sprd.rgb.edt.ply

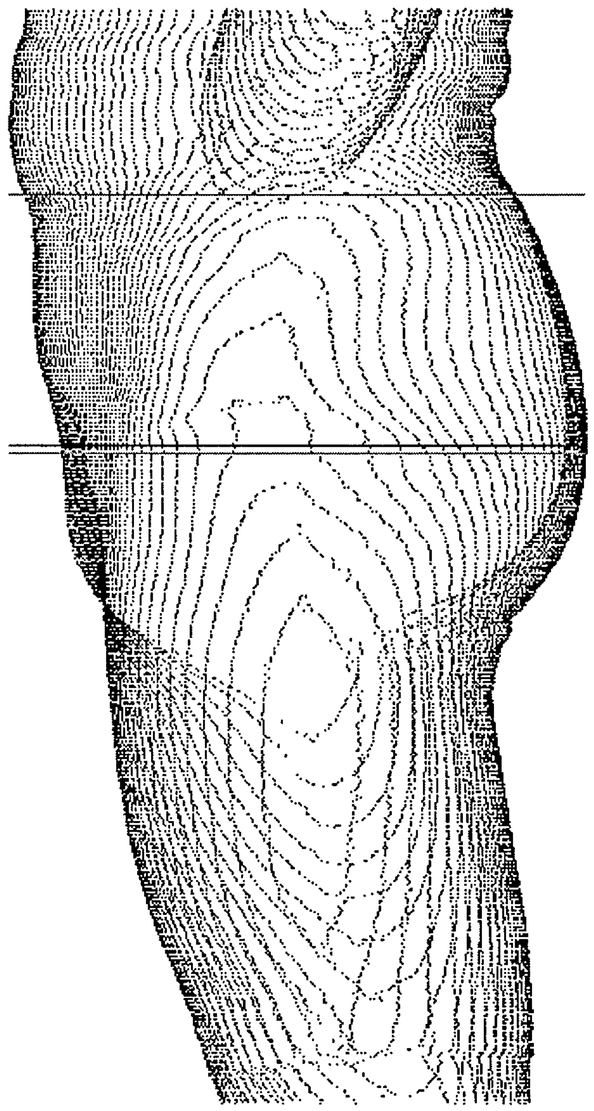
Waist Height Range

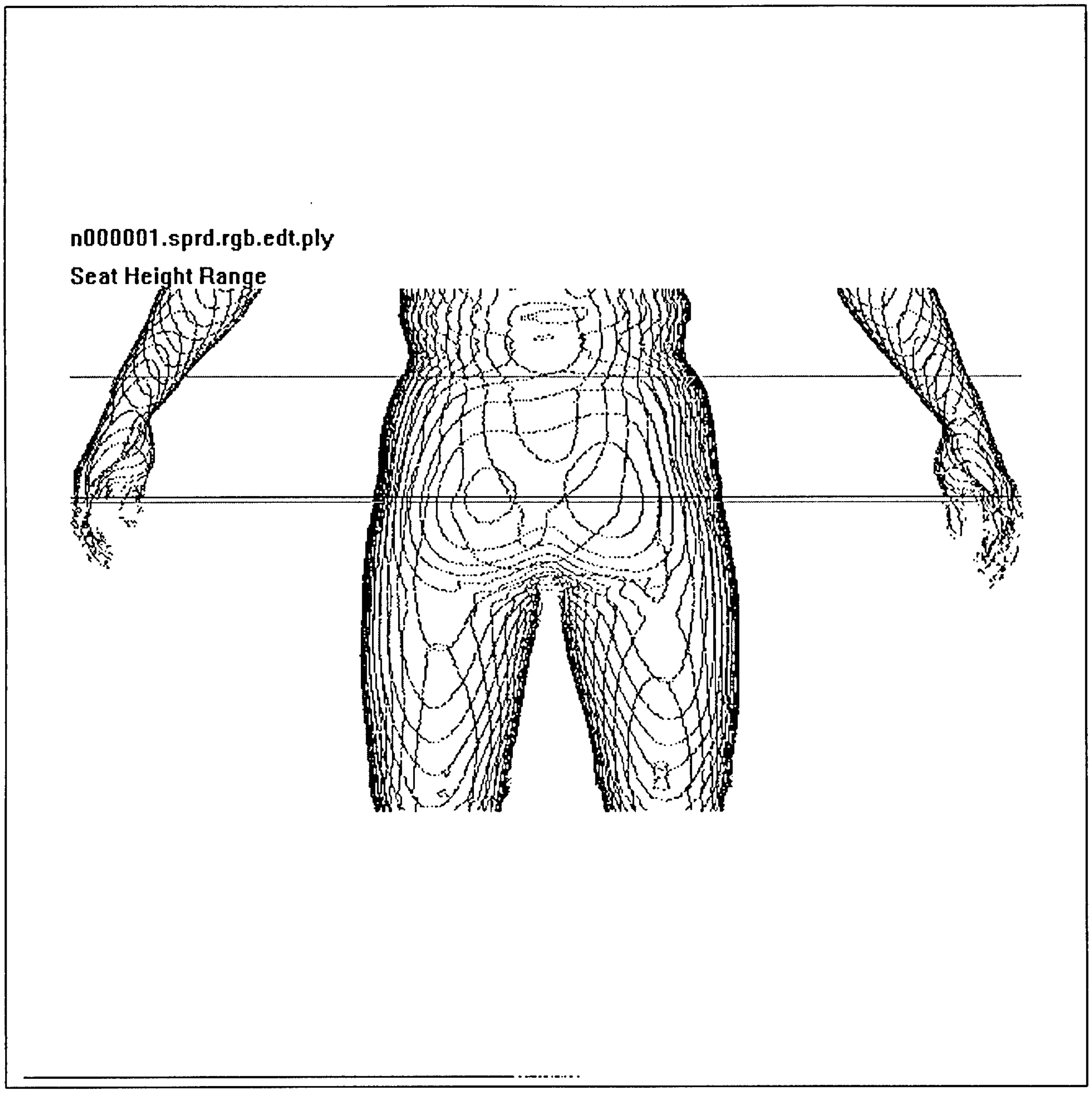
Waist = 31.084 " (789.523 ply units)

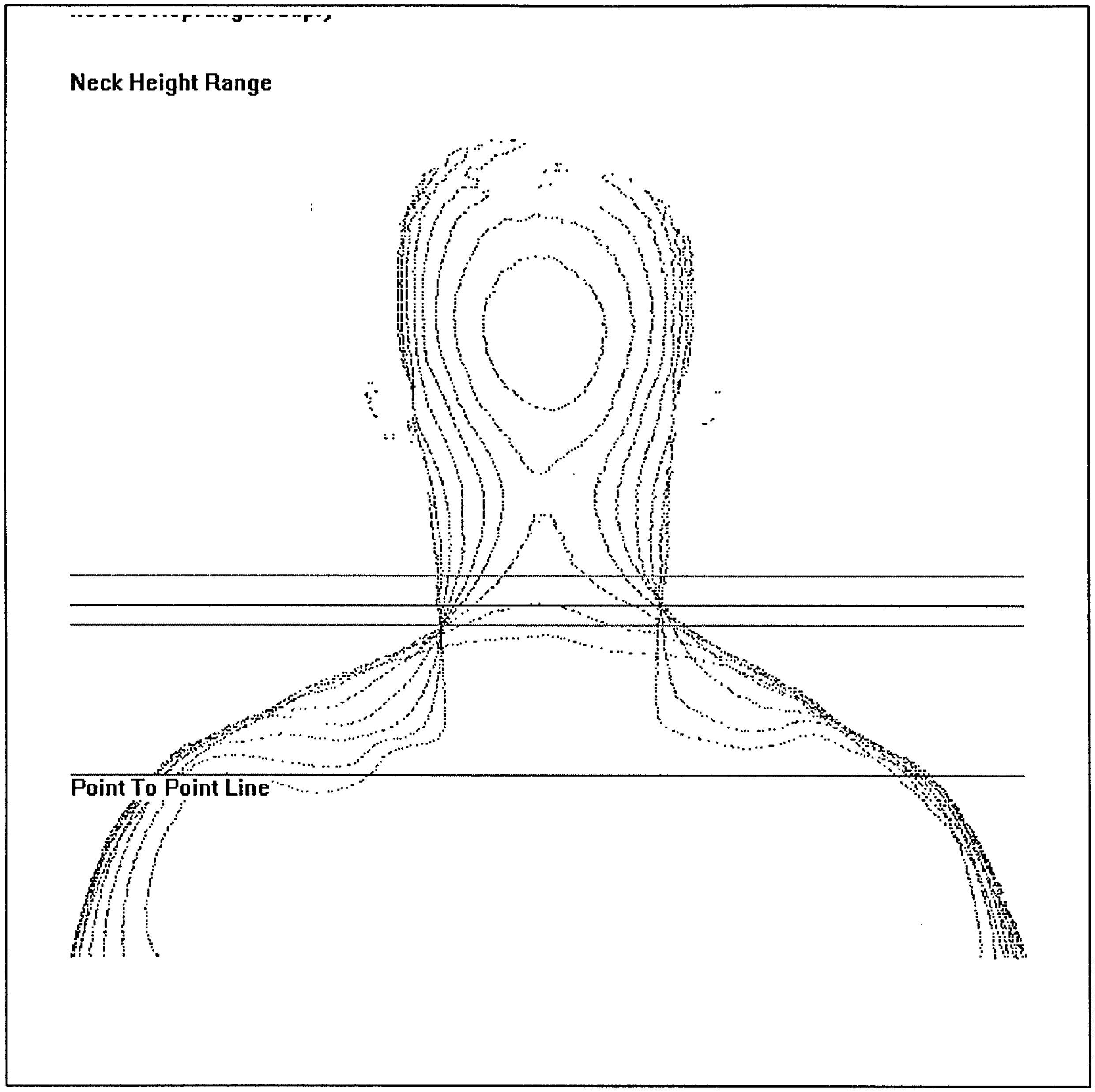

Top View N00001

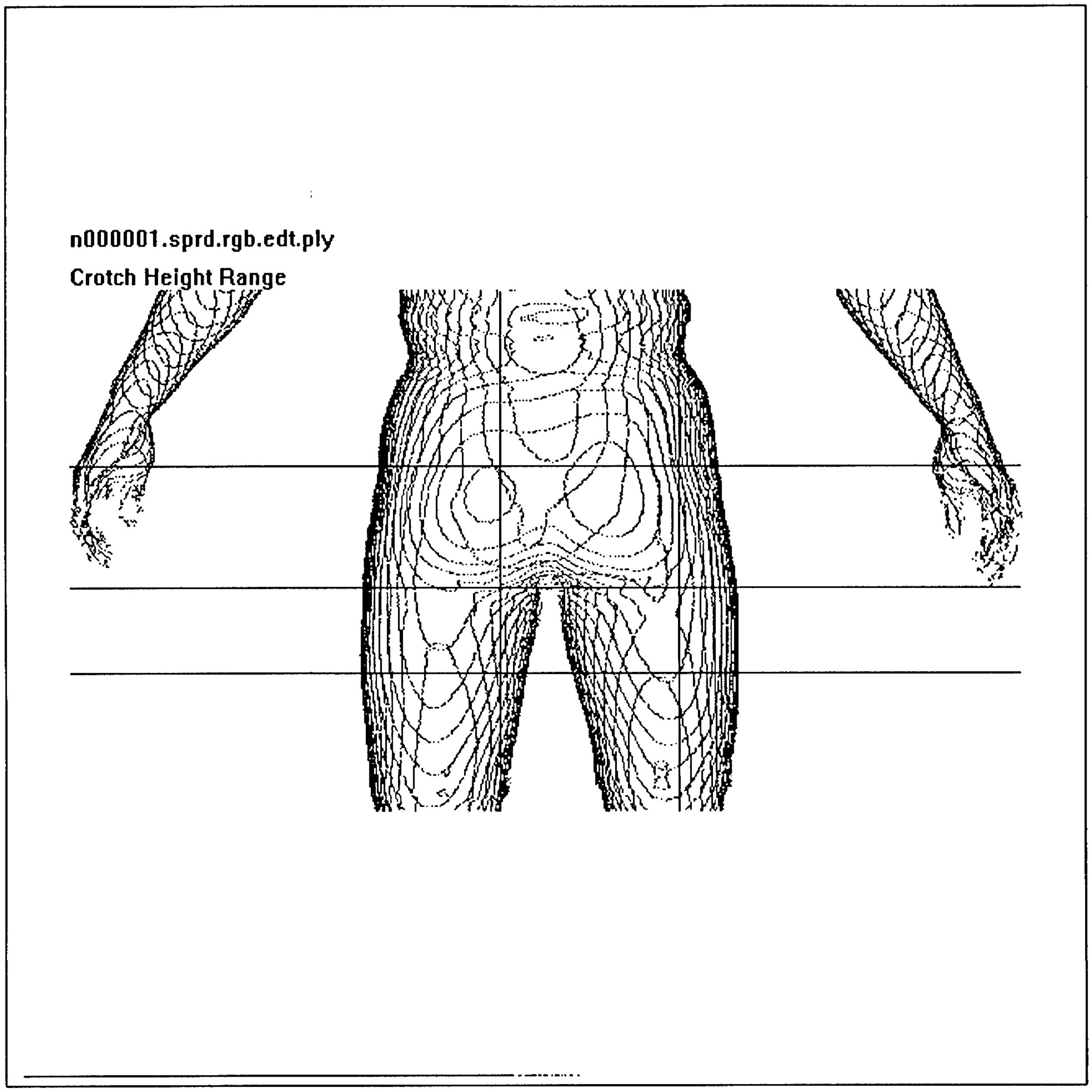

n000001.sprd.rgb.edt.ply

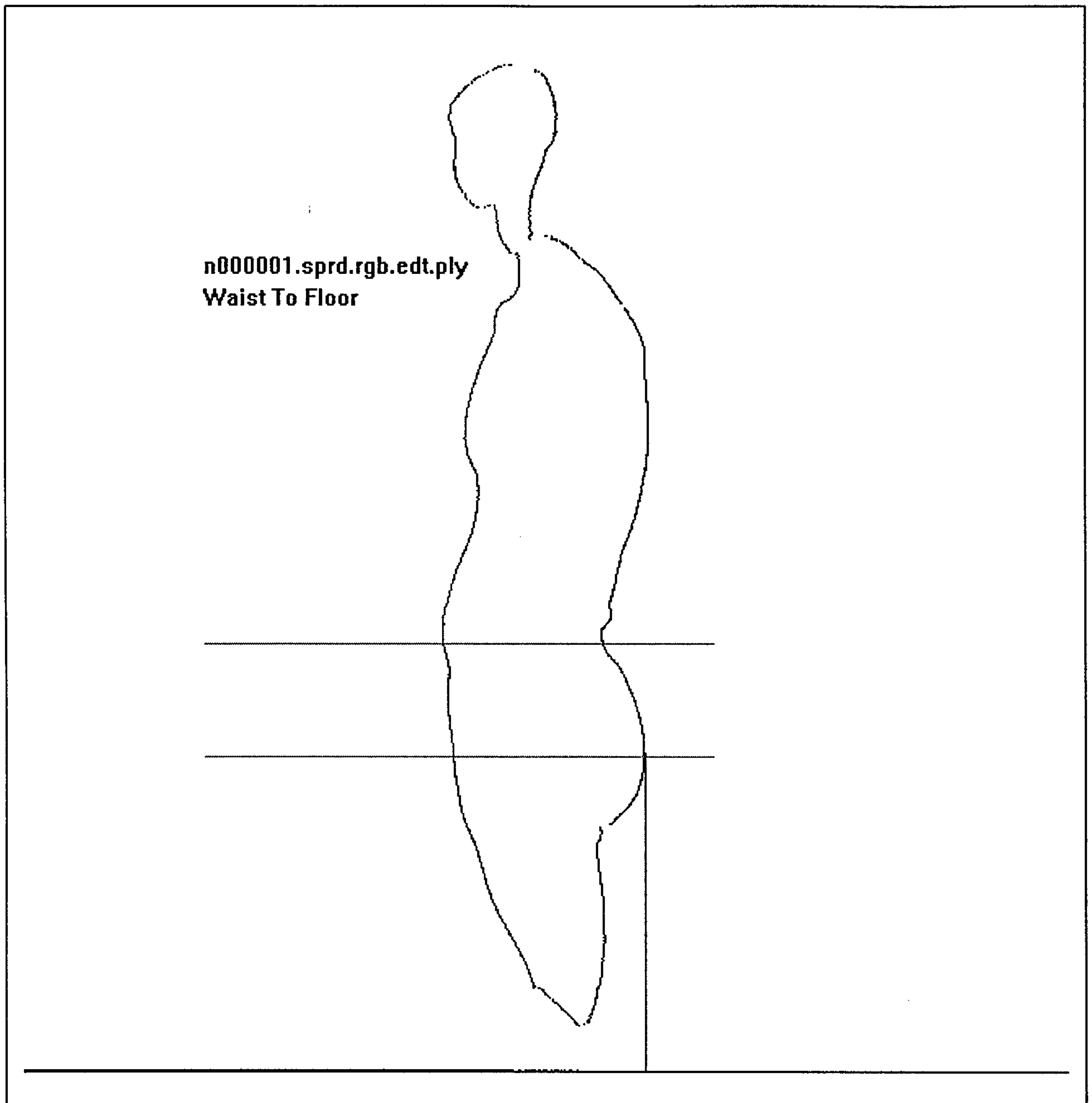
Seat Height Range



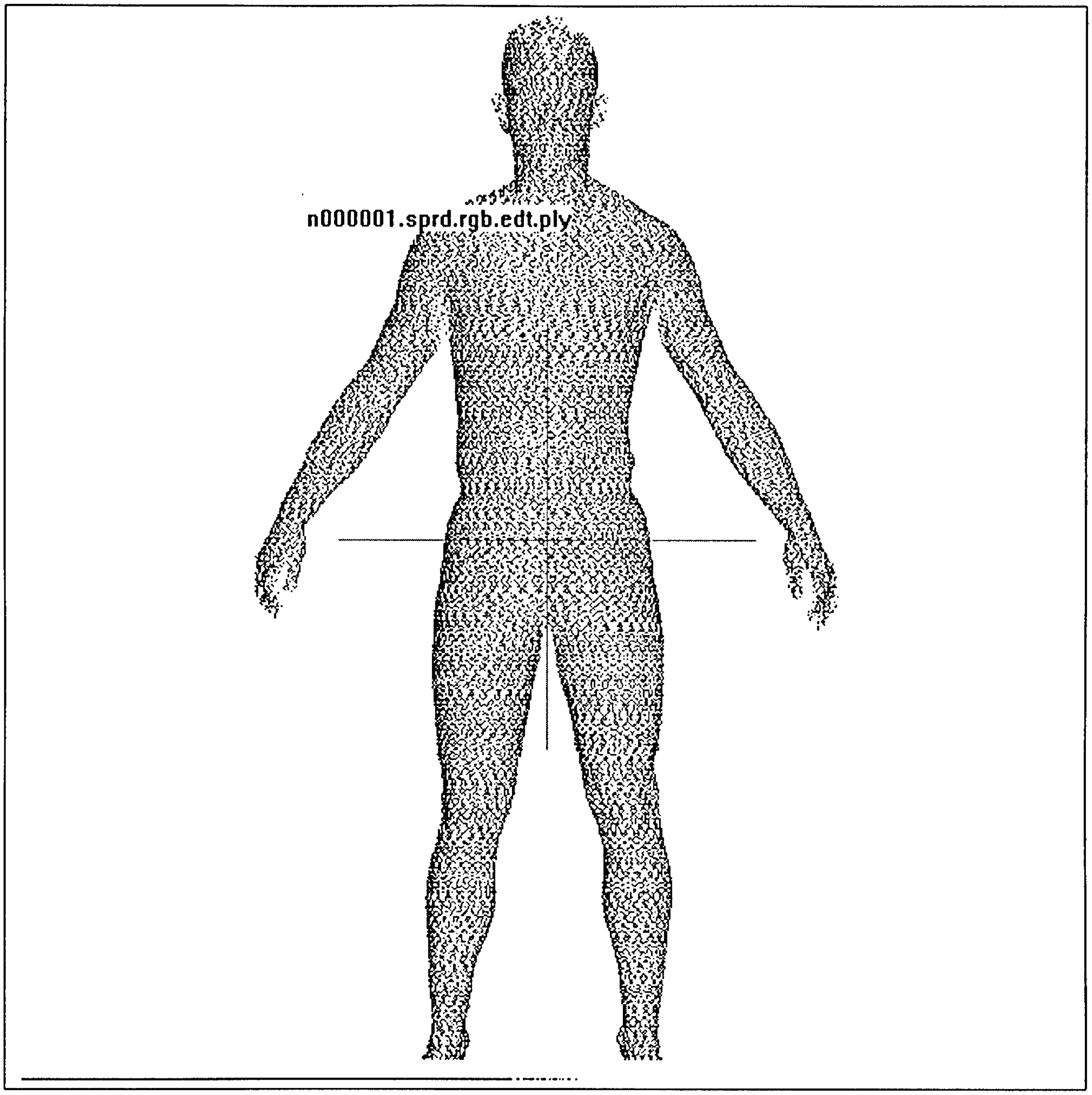

n000001.sprd.rgb.edt.ply
Seat Height Range

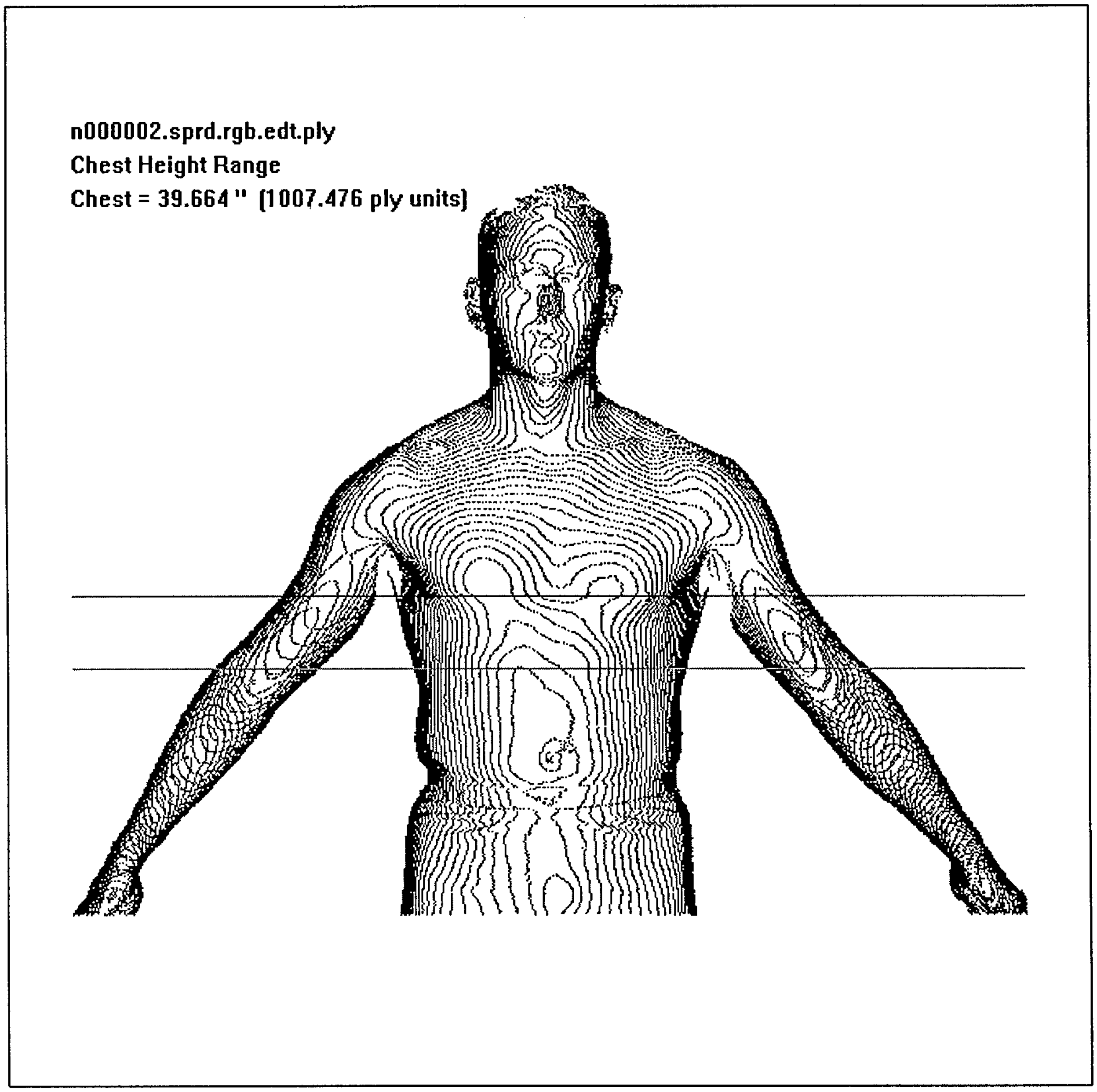

Neck Height Range
Point To Point Line

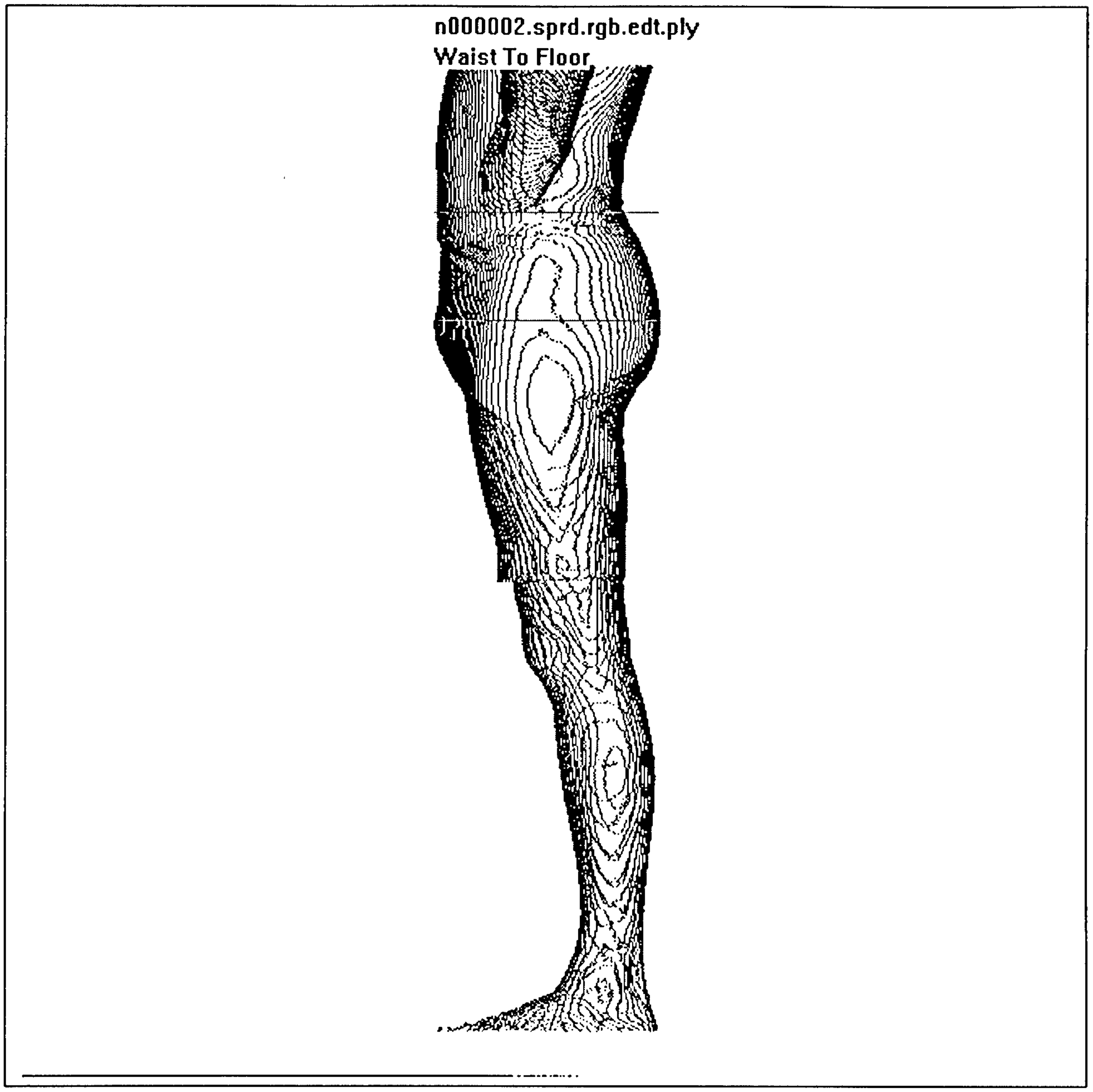

n000001.sprd.rgb.edt.ply
Crotch Height Range

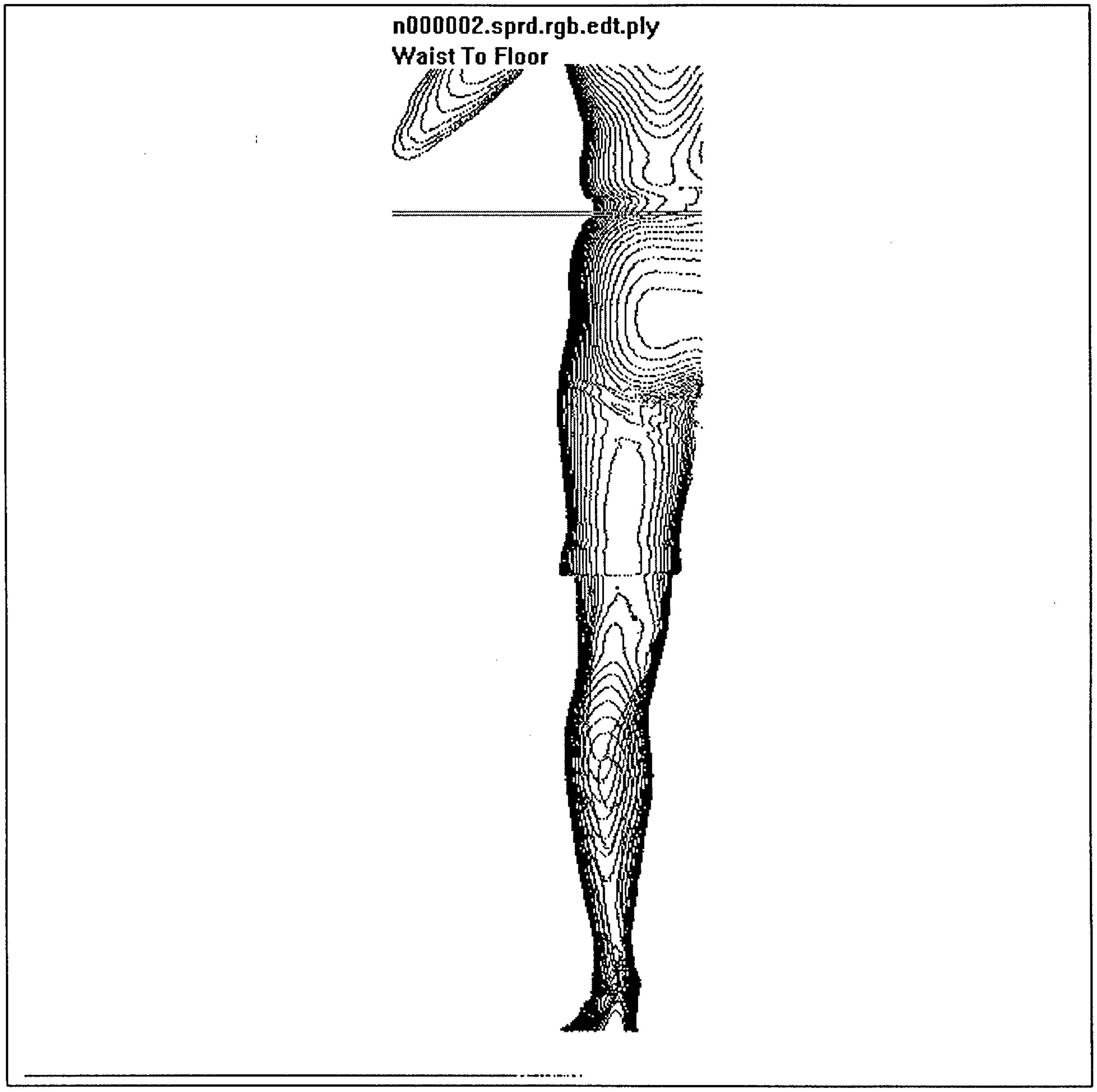


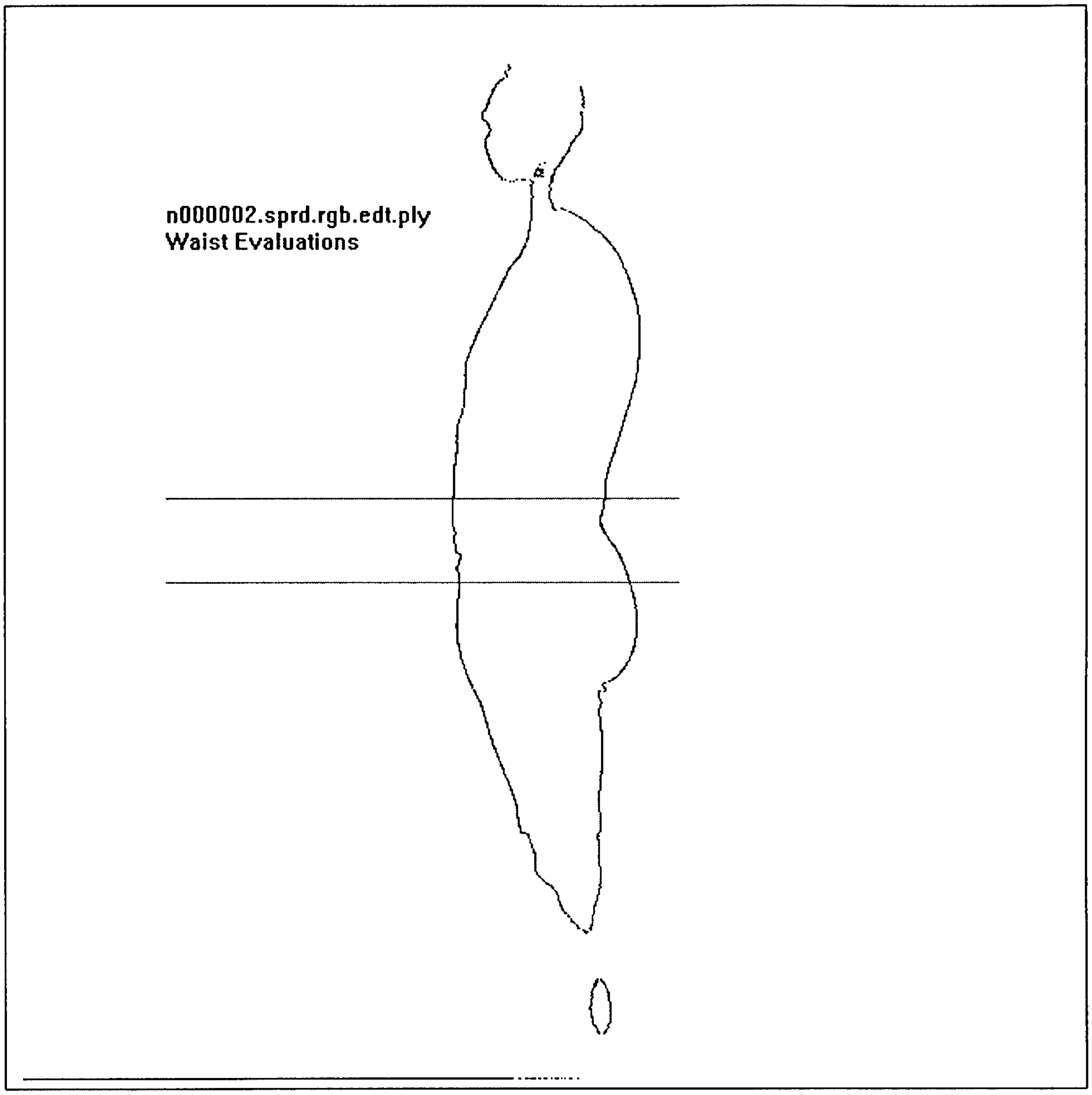
Back-to-floor measurement

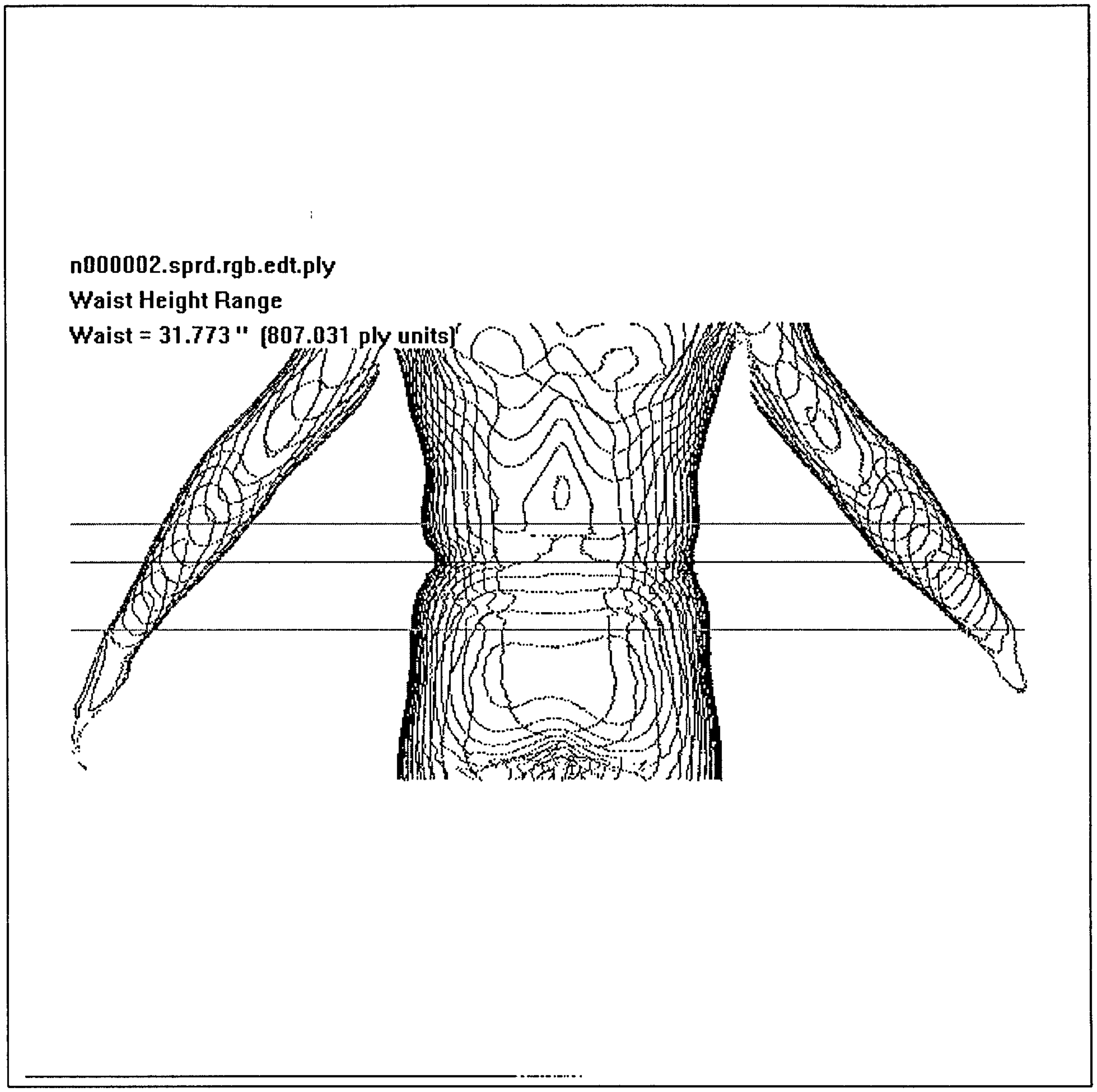

n000001.sprd.rgb.edt.ply

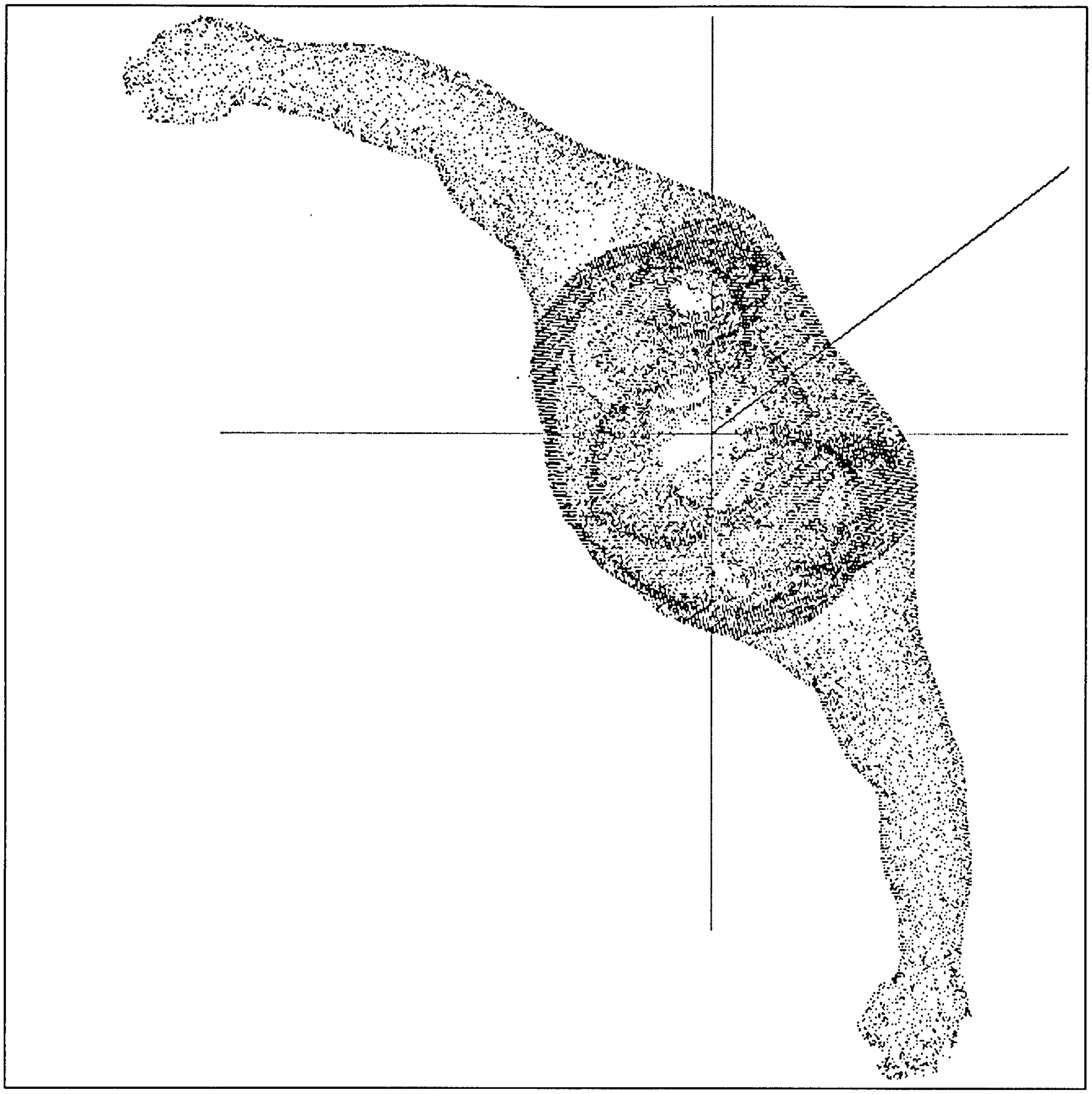

n000002.sprd.rgb.edt.ply
Chest Height Range
Chest = 39.664 " (1007.476 ply units)

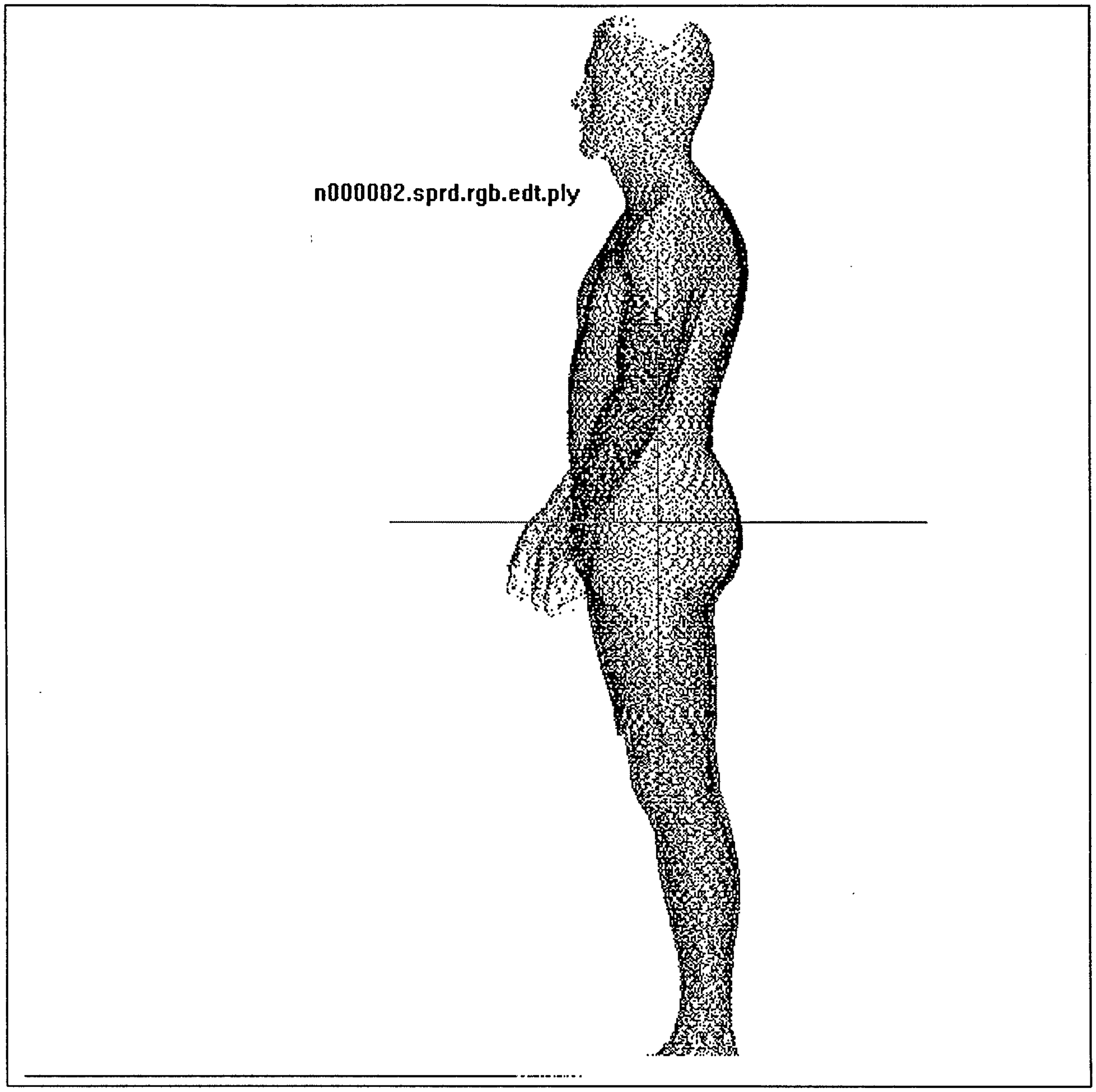

n000002.sprd.rgb.edt.ply
Waist To Floor

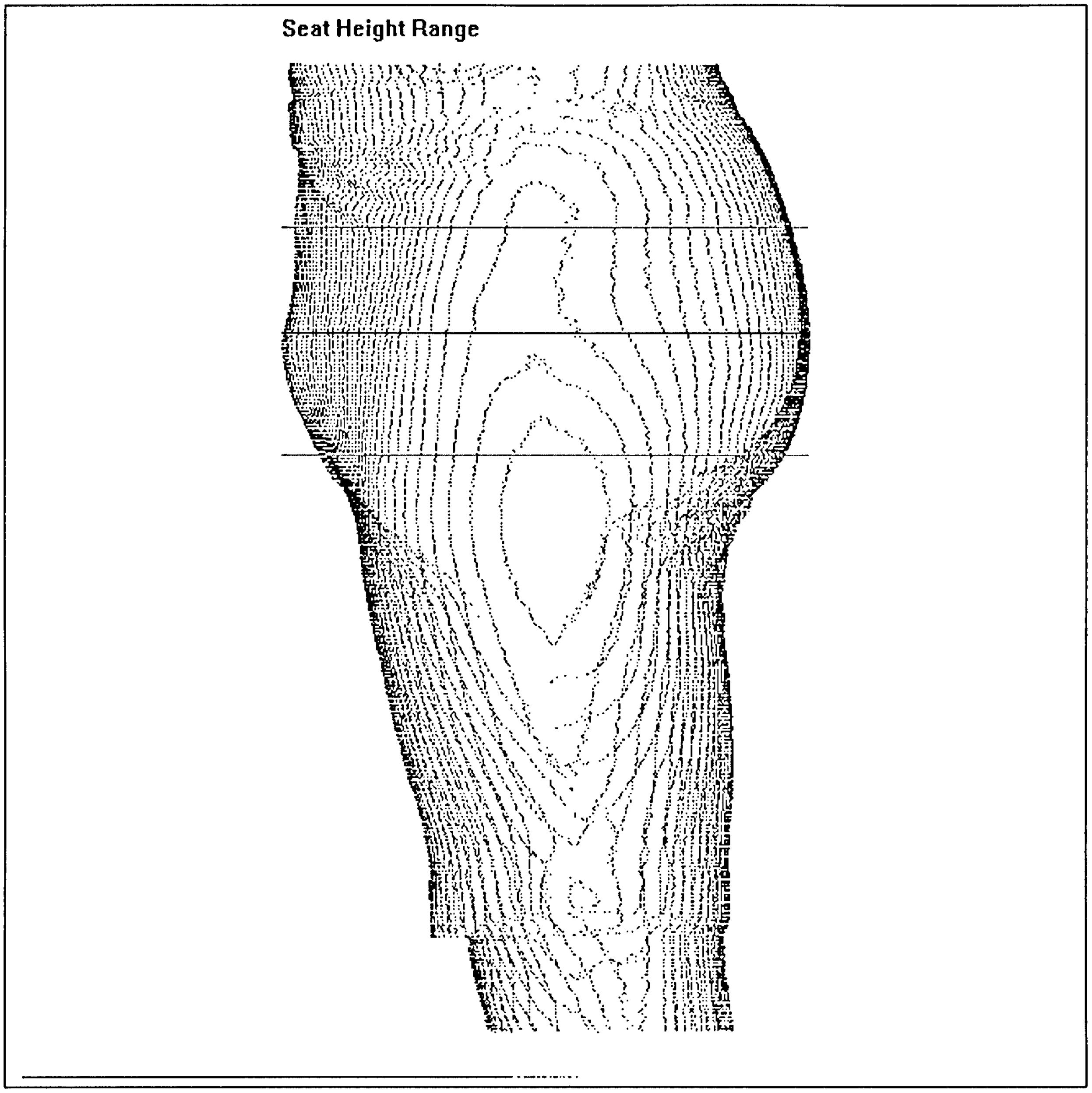

n000002.sprd.rgb.edt.ply
Waist To Floor

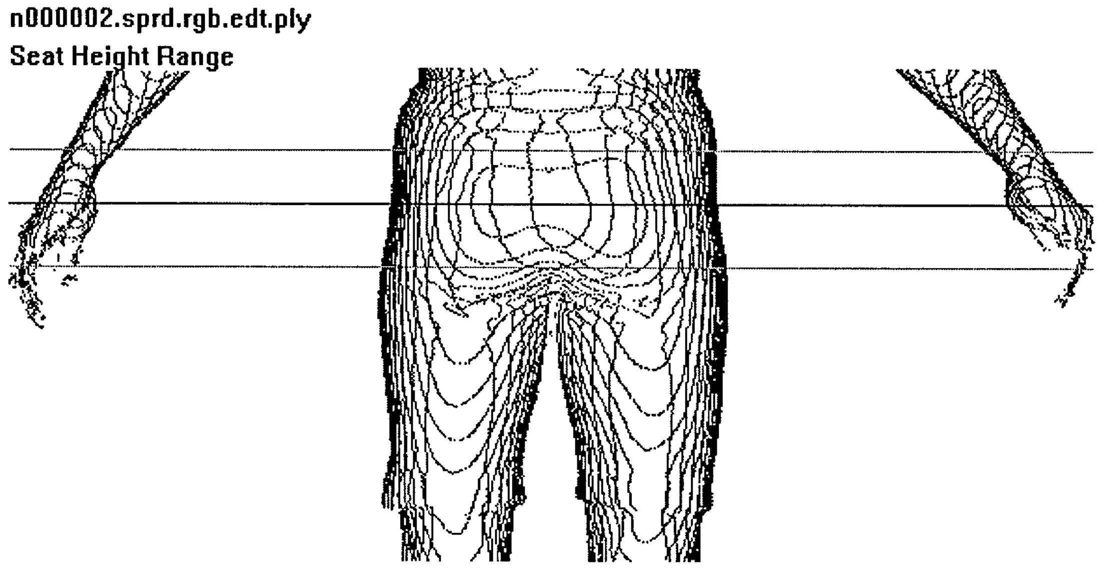

n000002.sprd.rgb.edt.ply
Waist Evaluations

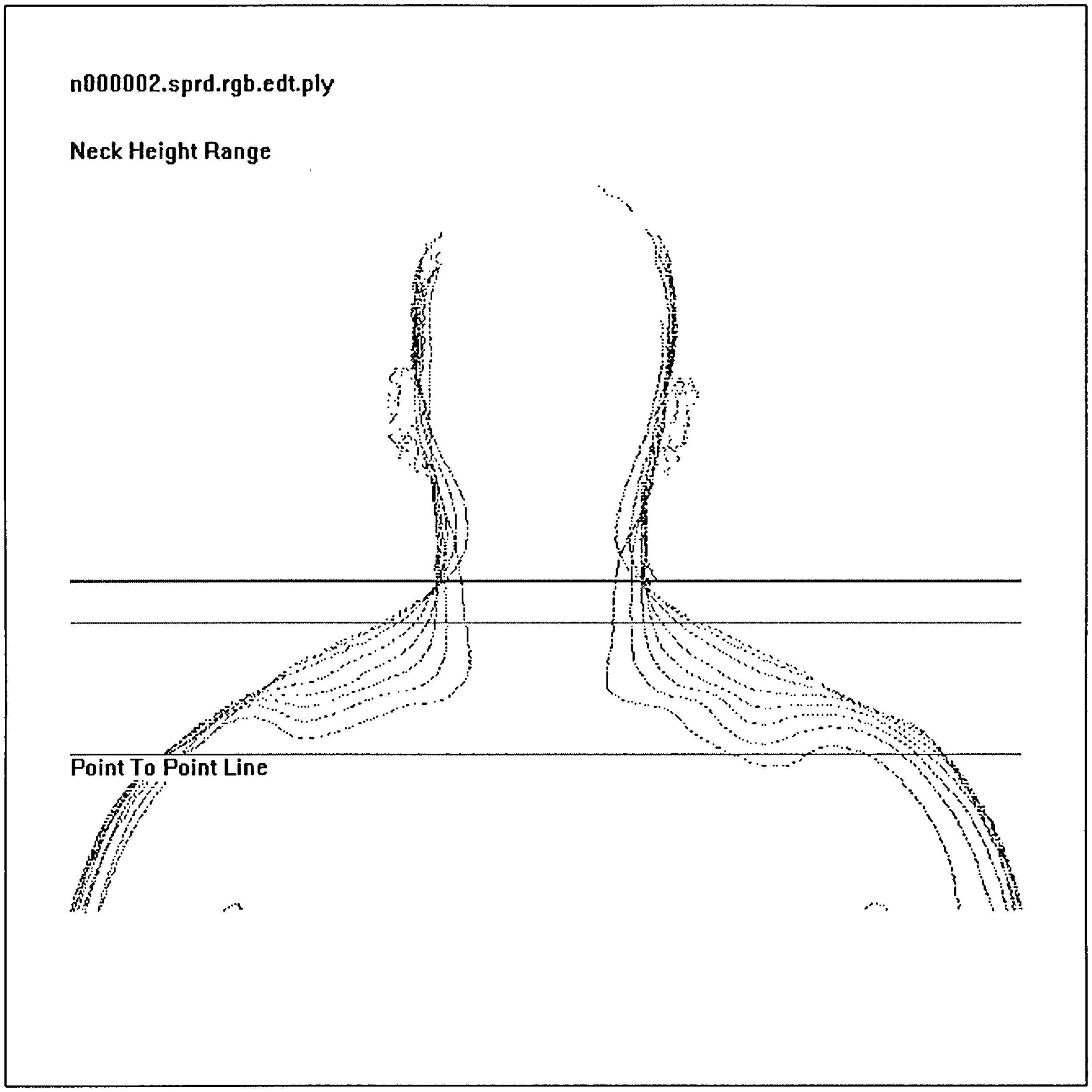

n000002.sprd.rgb.edt.ply
Waist Height Range
Waist = 31.773 " (807.031 ply units)

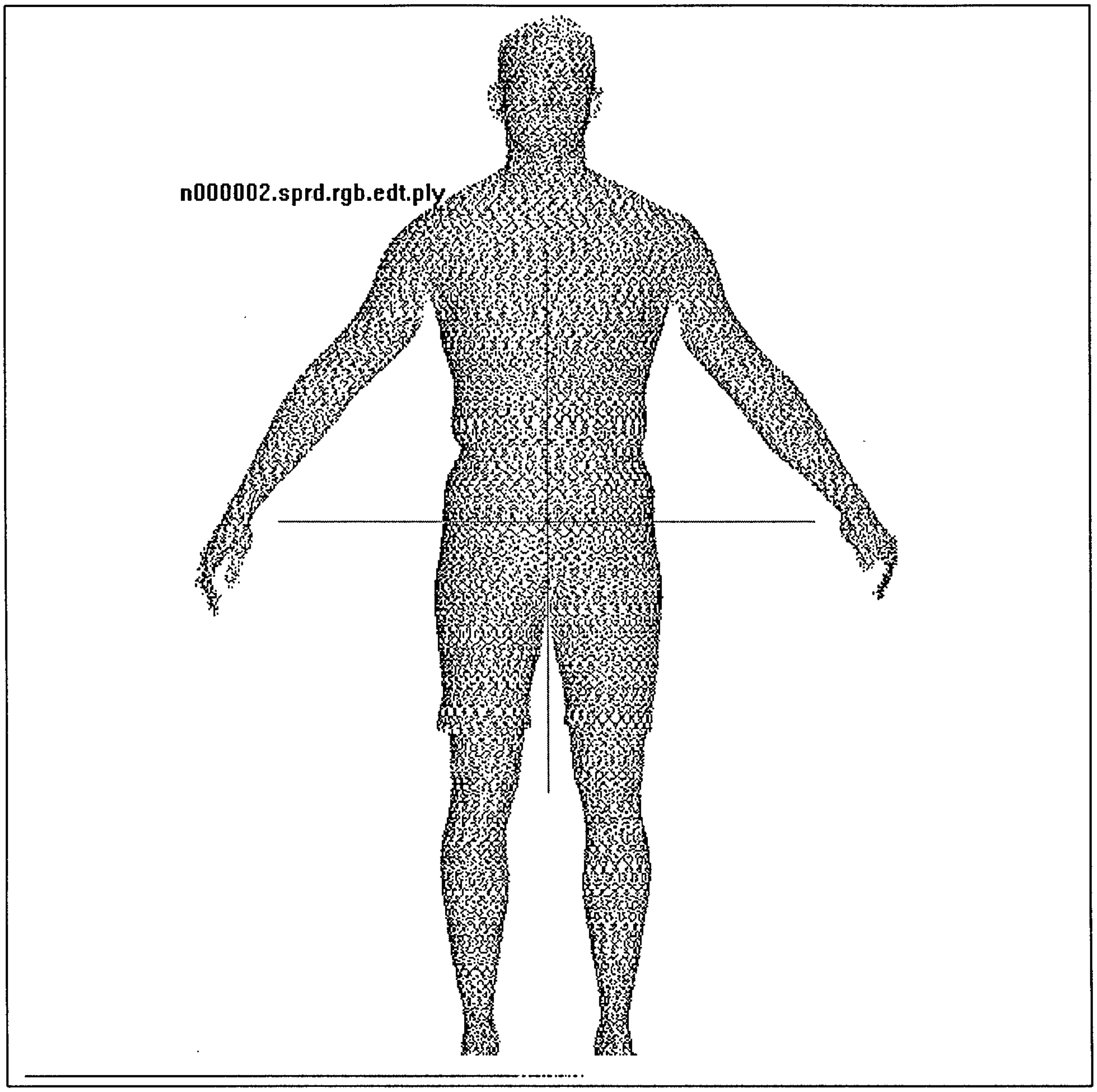


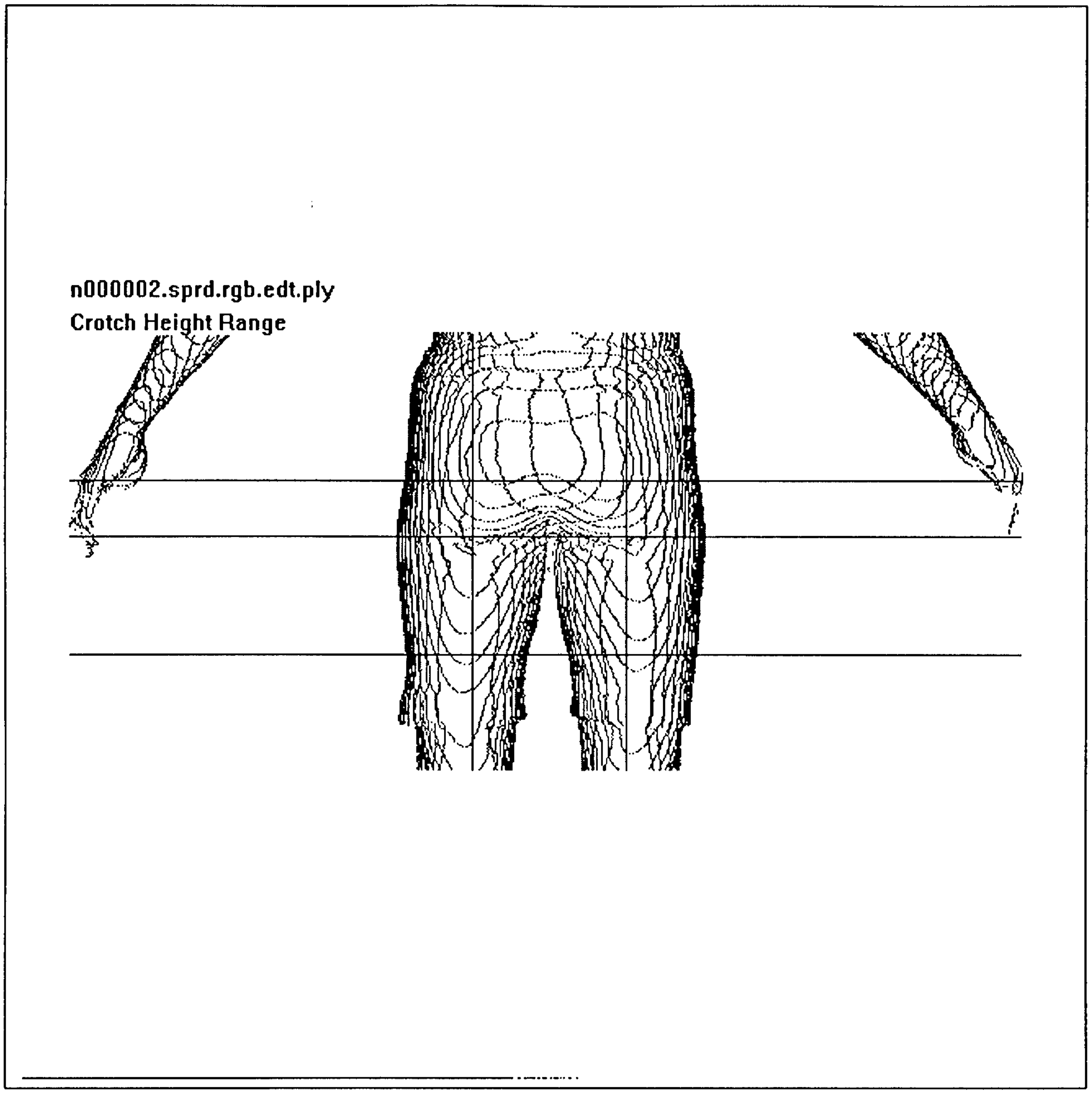

n000002.sprd.rgb.edt.ply

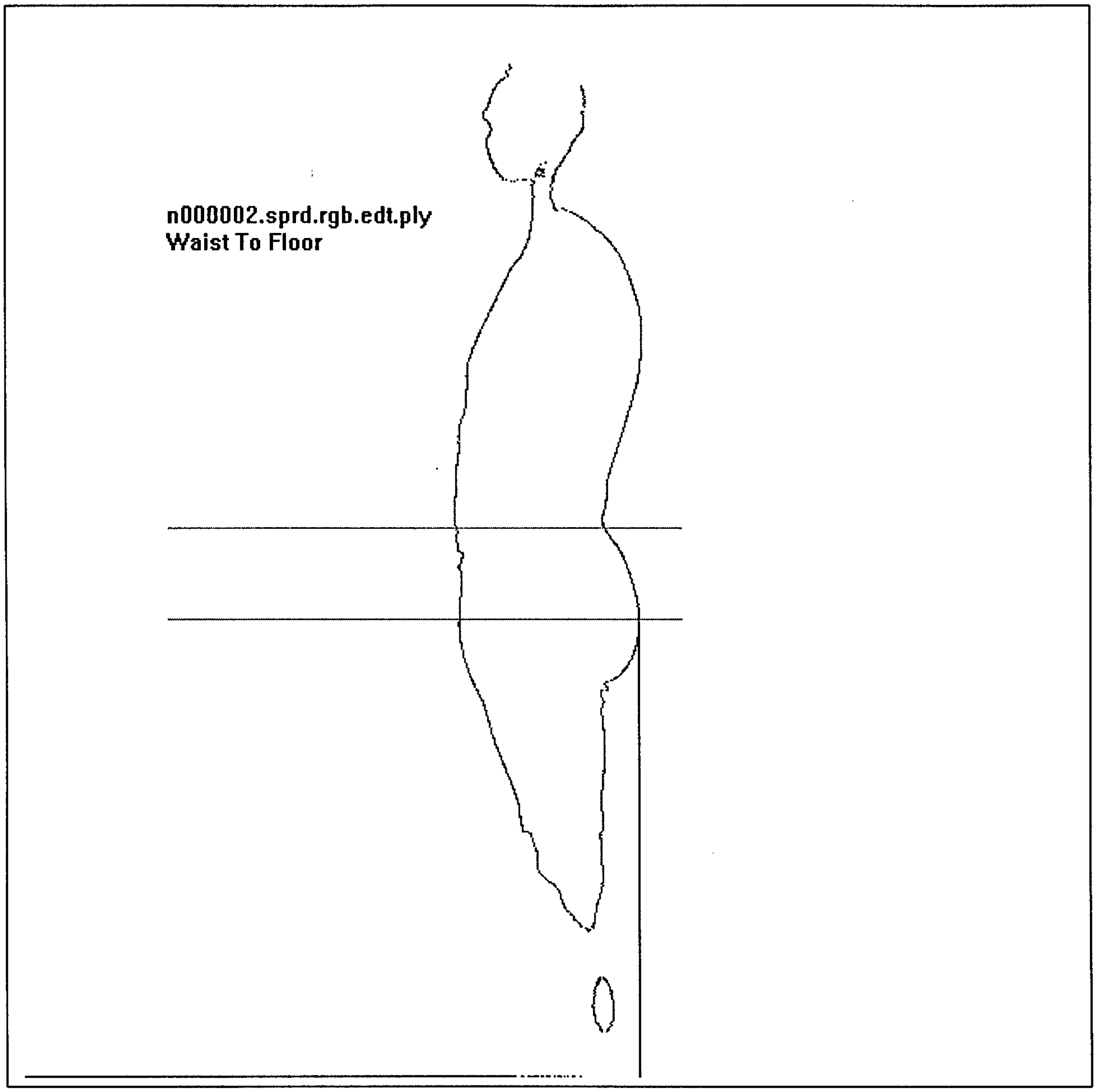

Seat Height Range

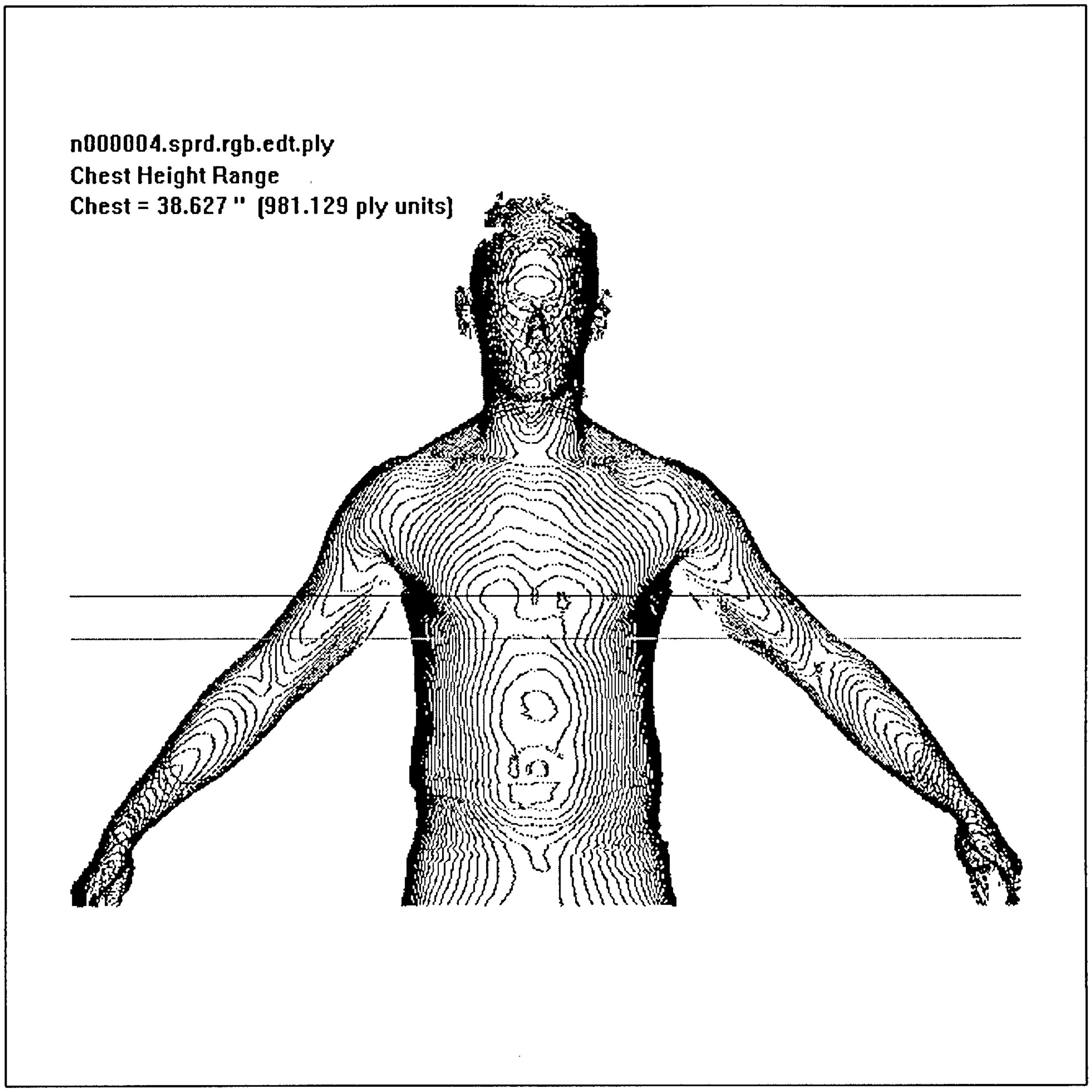

n000002.sprd.rgb.edt.ply
Seat Height Range


n000002.sprd.rgb.edt.ply
Neck Height Range
Point To Point Line

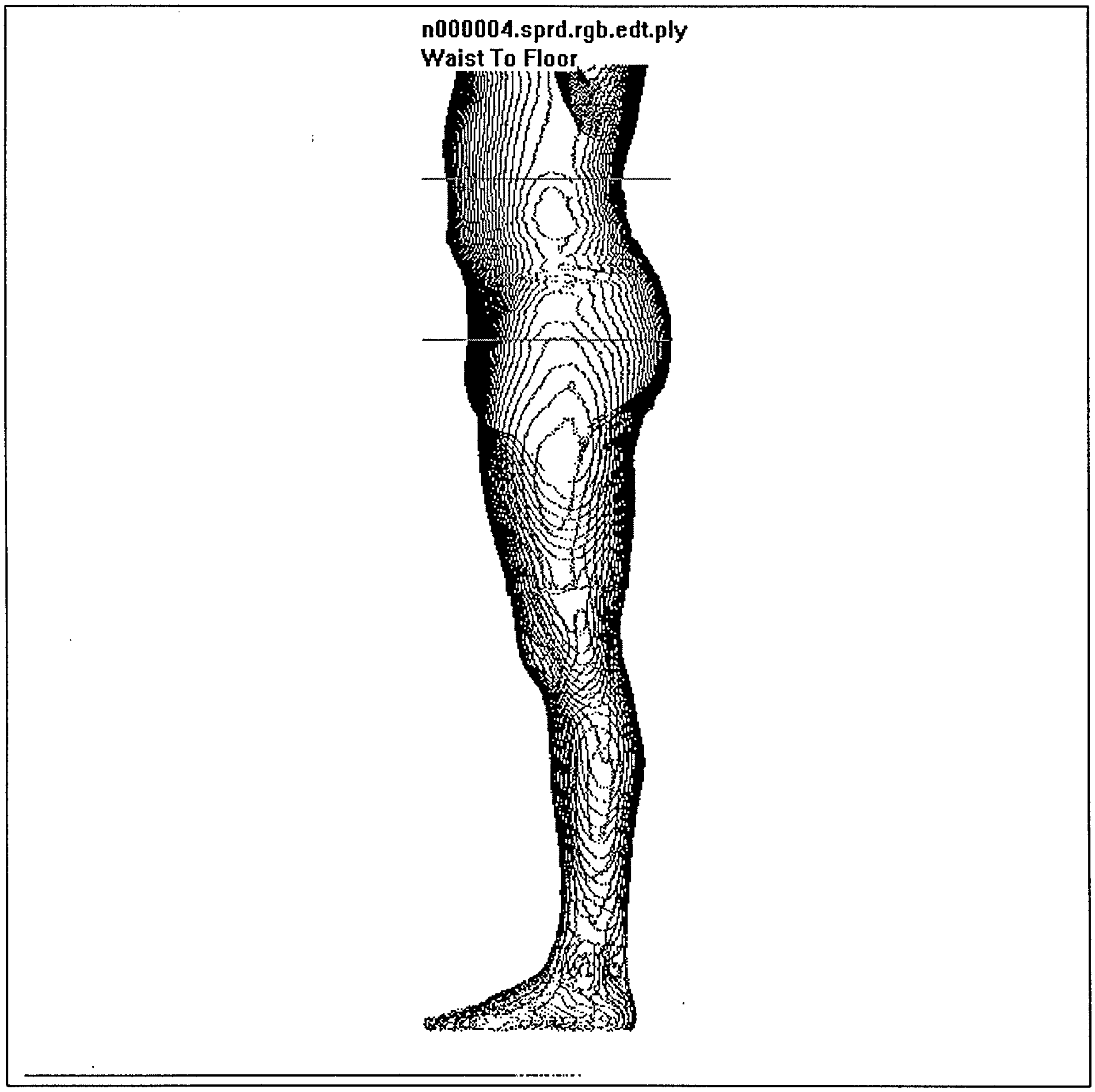

n000002.sprd.rgb.edt.ply

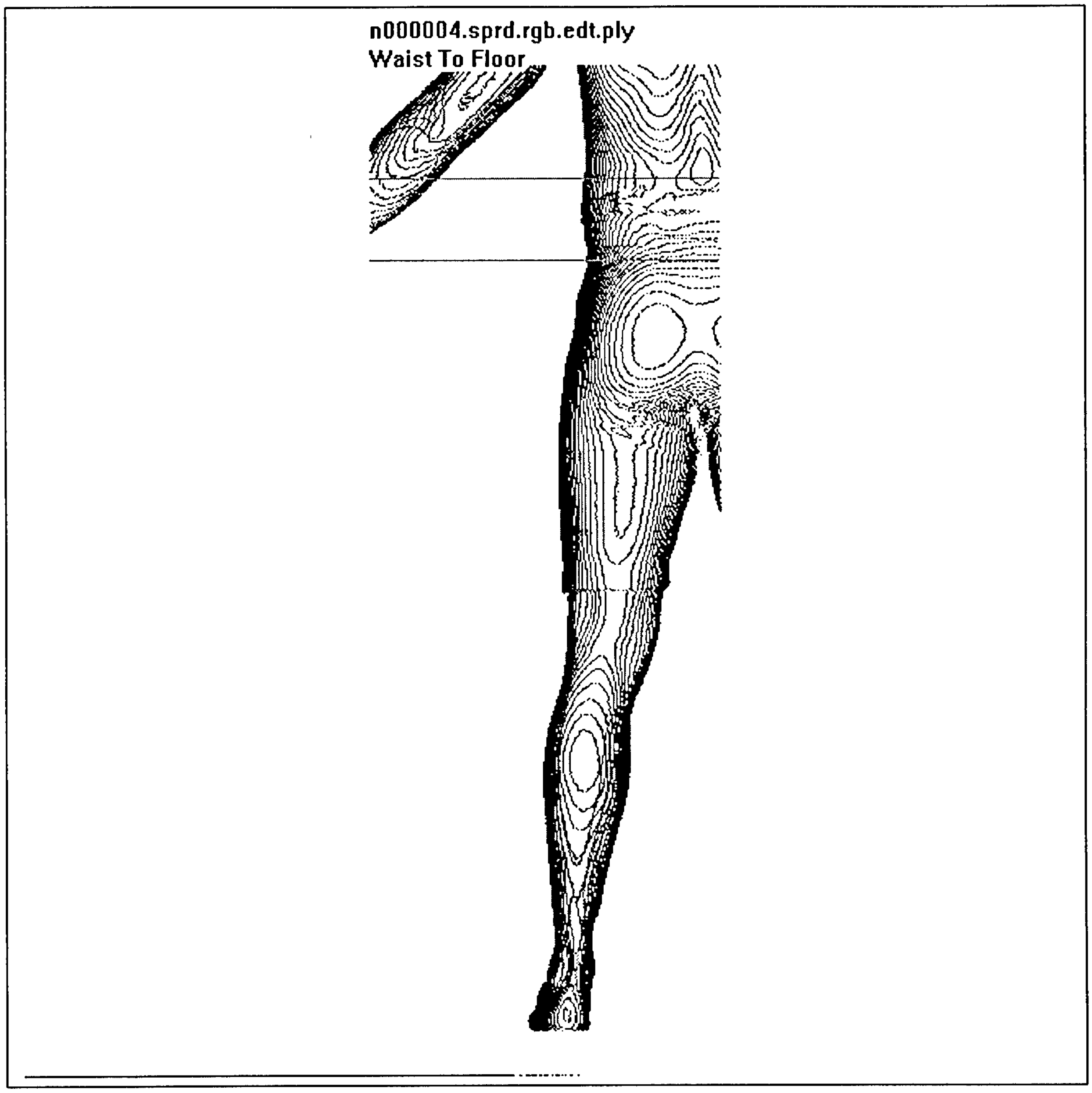

n000002.sprd.rgb.edt.ply
Crotch Height Range

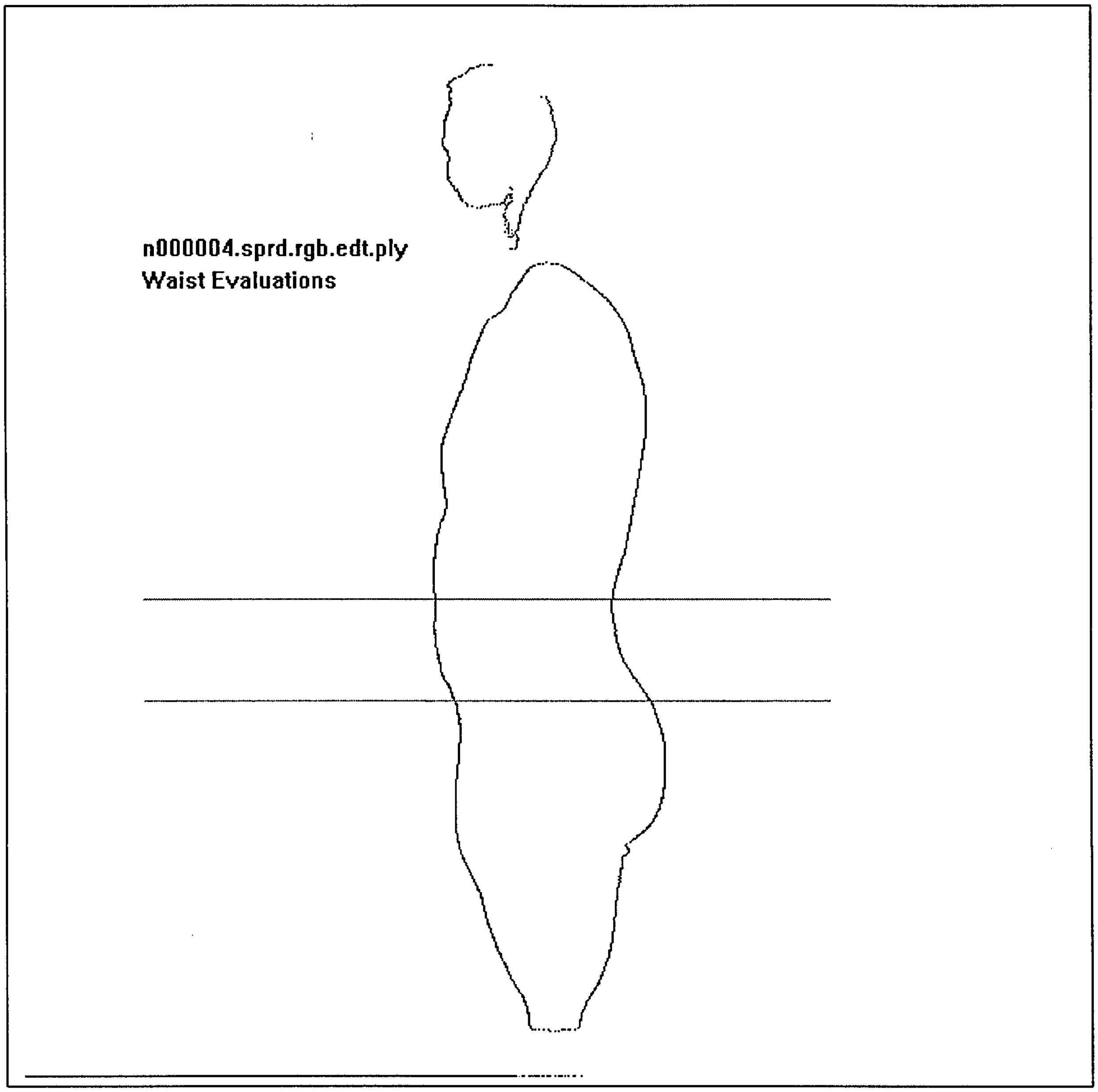

n000002.sprd.rgb.edt.ply
Waist To Floor

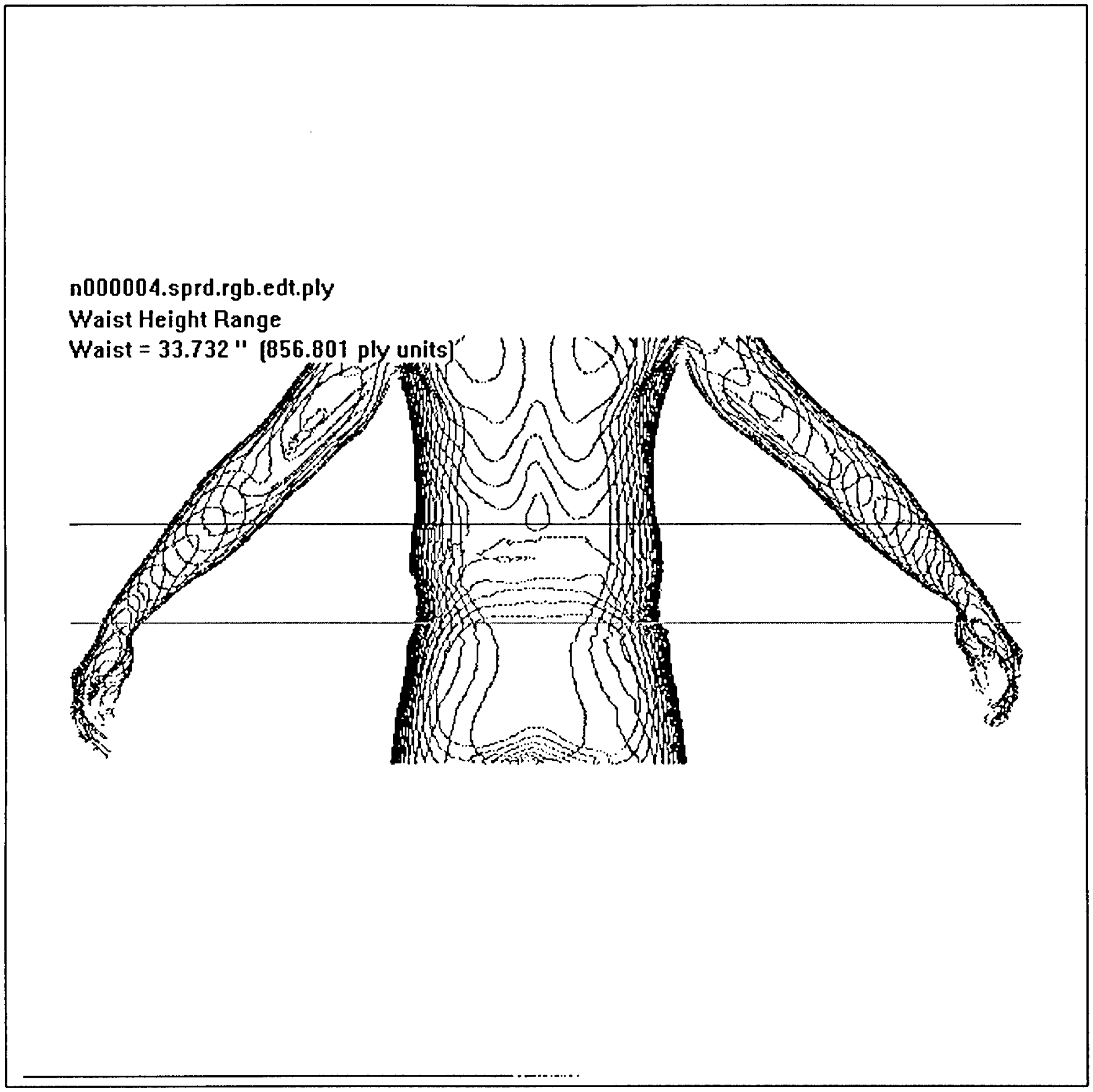


n000004.sprd.rgb.edt.ply
Chest Height Range
Chest = 38.627 " (981.129 ply units)

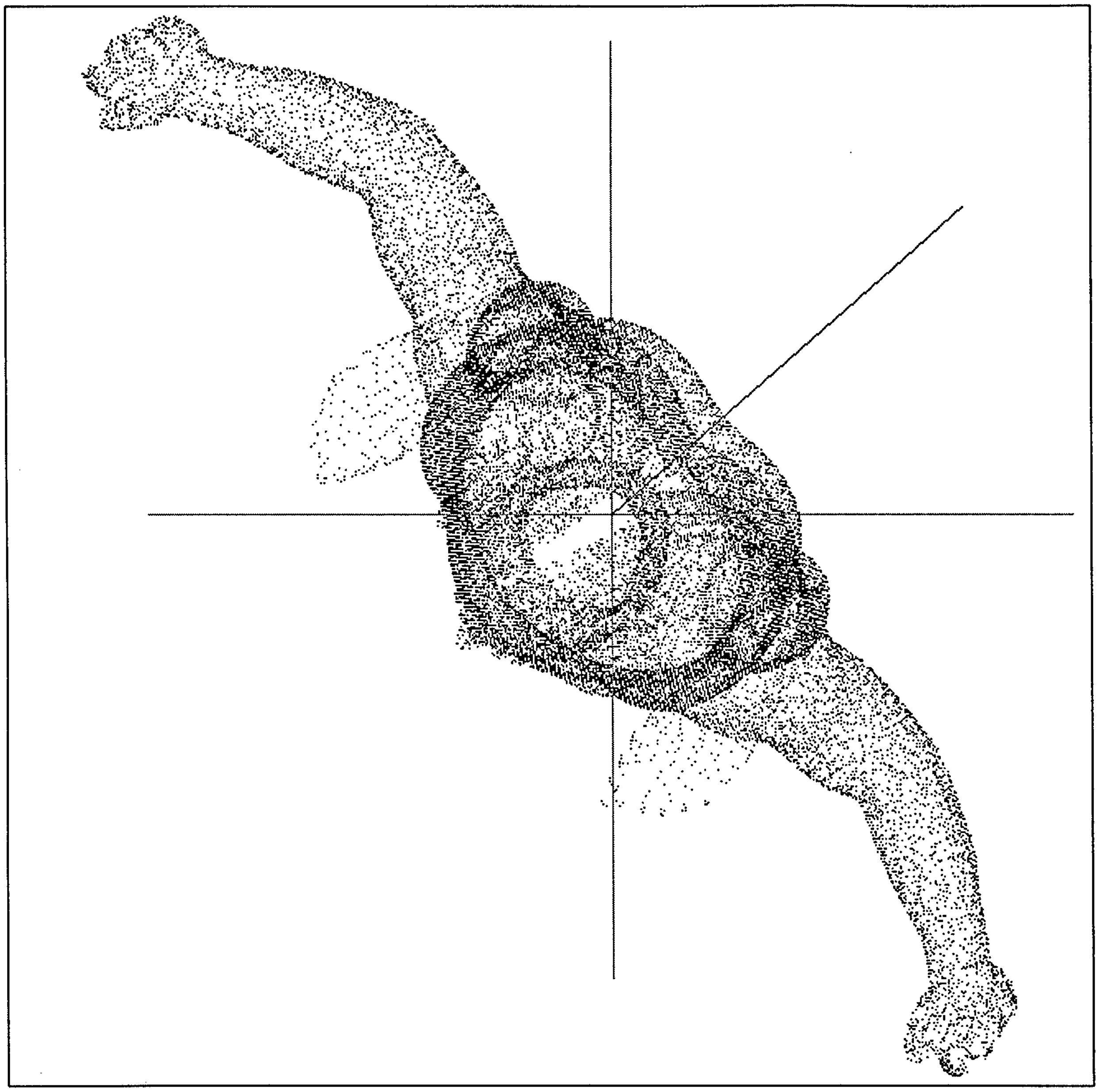

n000004.sprd.rgb.edt.ply
Waist To Floor

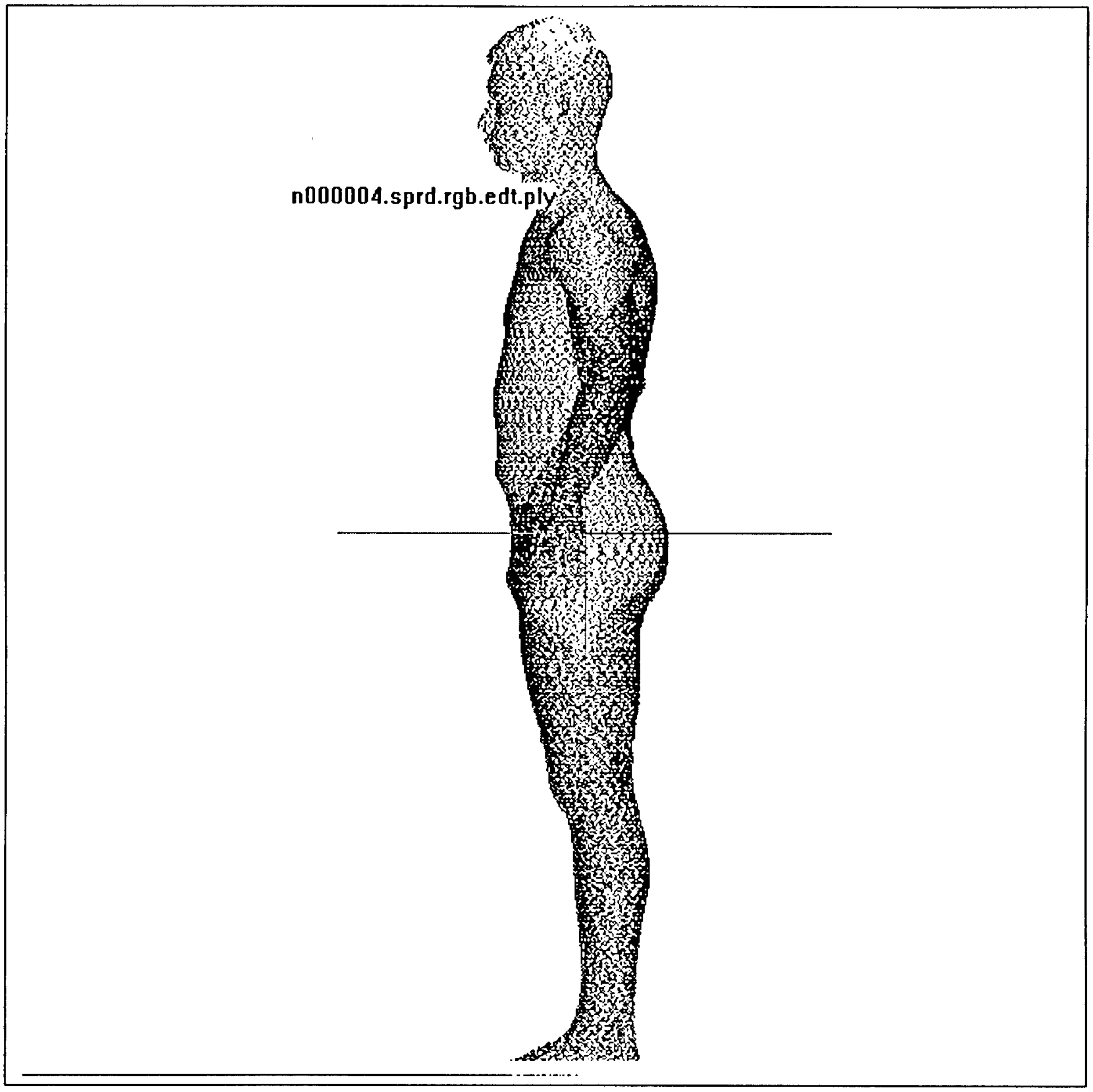

n000004.sprd.rgb.edt.ply
Waist To Floor


n000004.sprd.rgb.edt.ply
Waist Evaluations

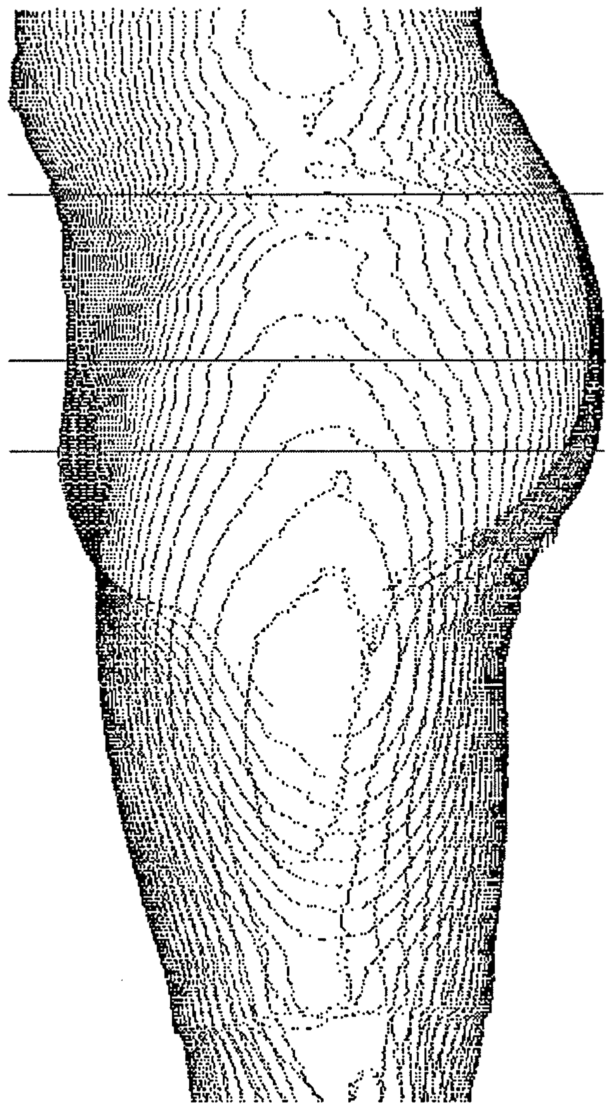


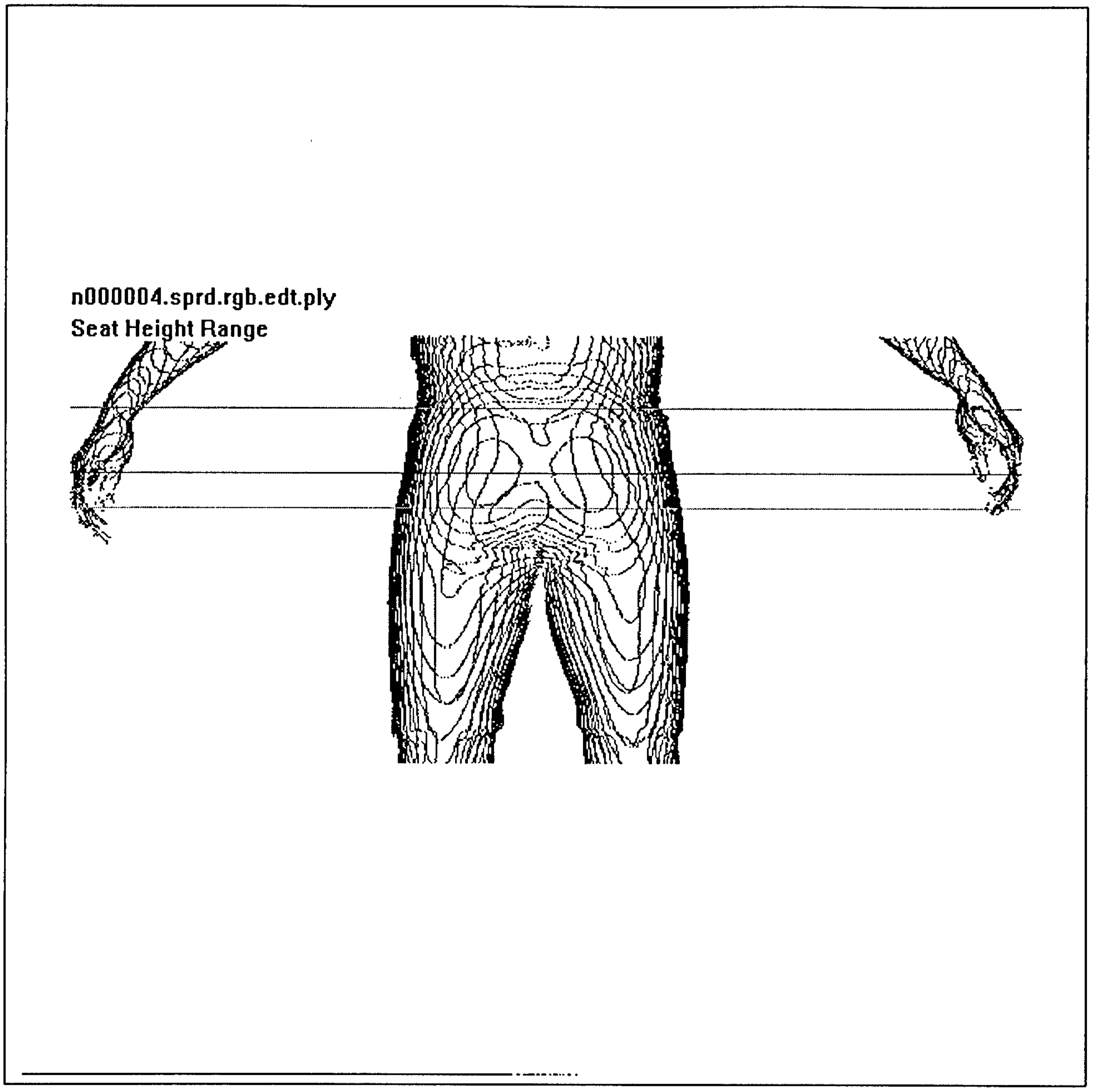
n000004.sprd.rgb.edt.ply
Waist Height Range
Waist = 33.732 " (856.801 ply units)

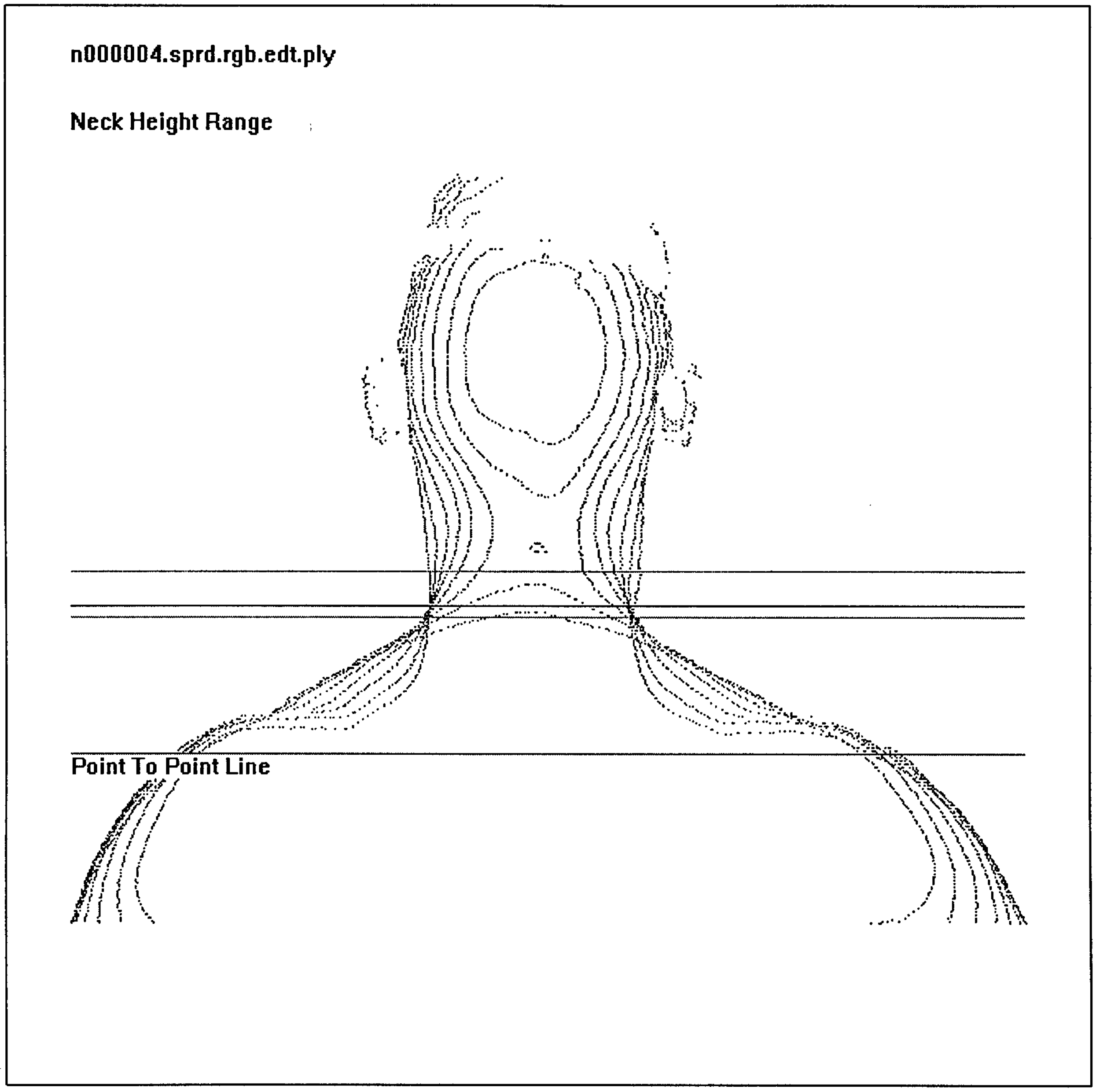


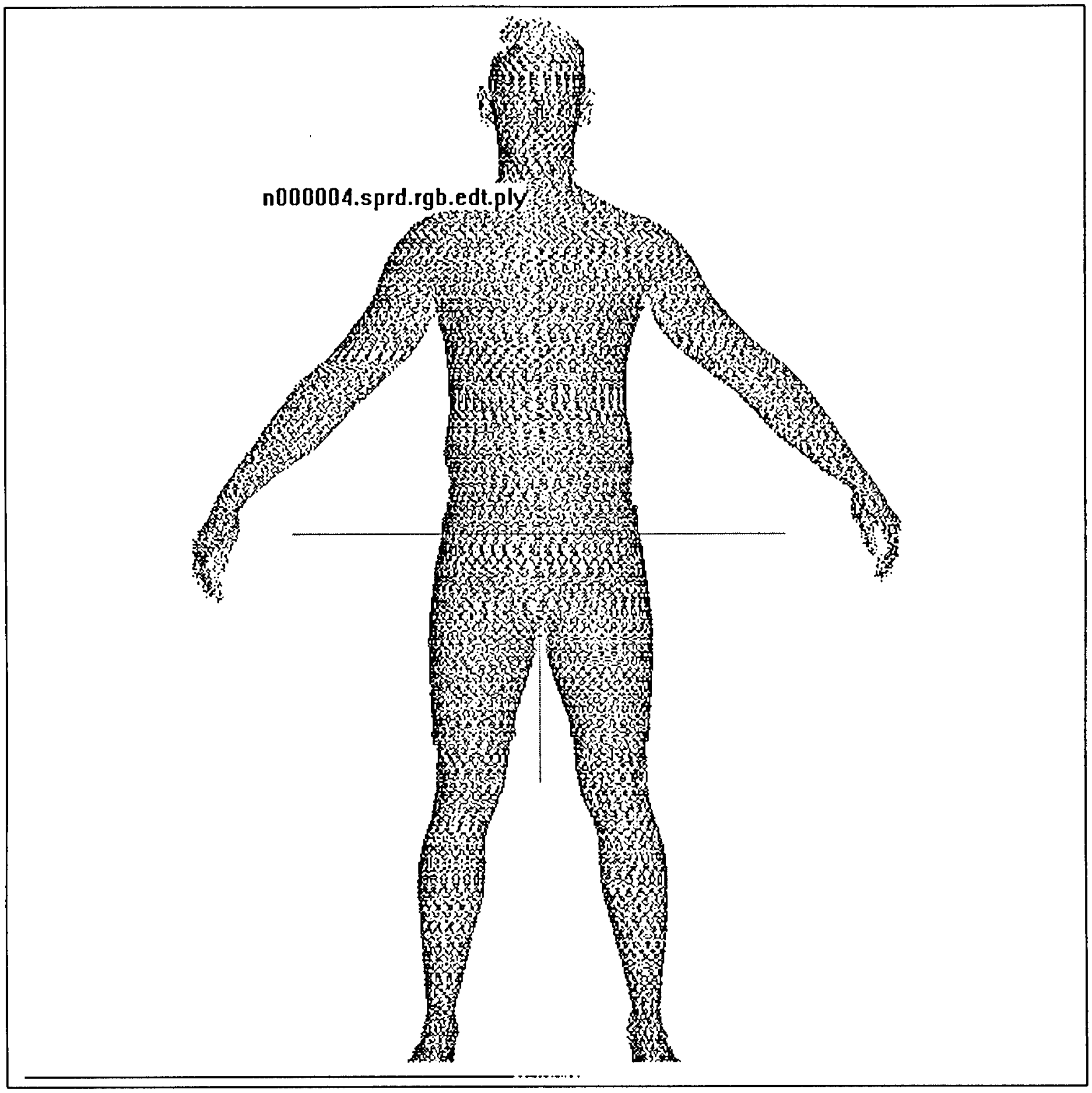

n000004.sprd.rgb.edt.ply

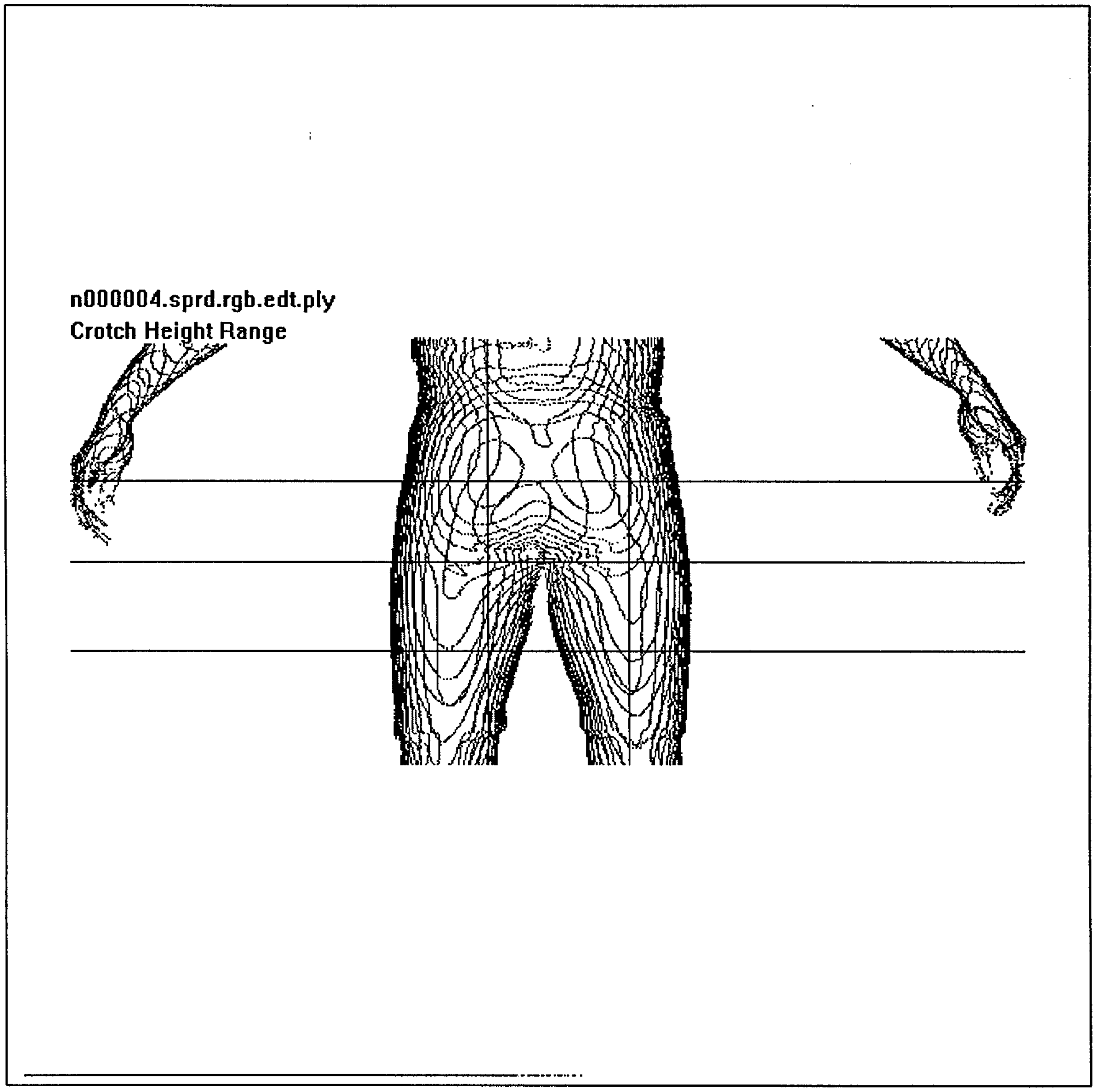
Seat Height Range

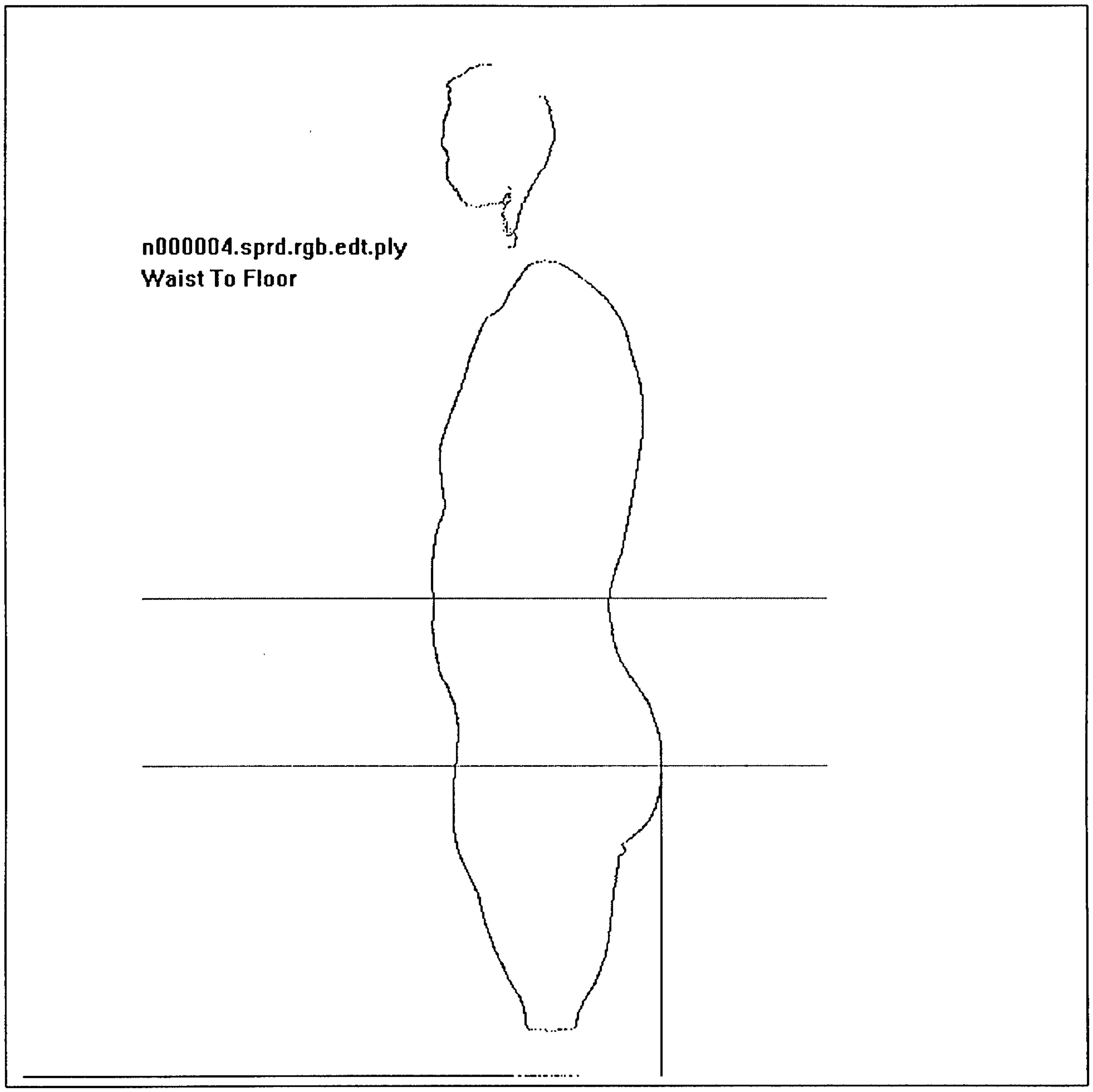


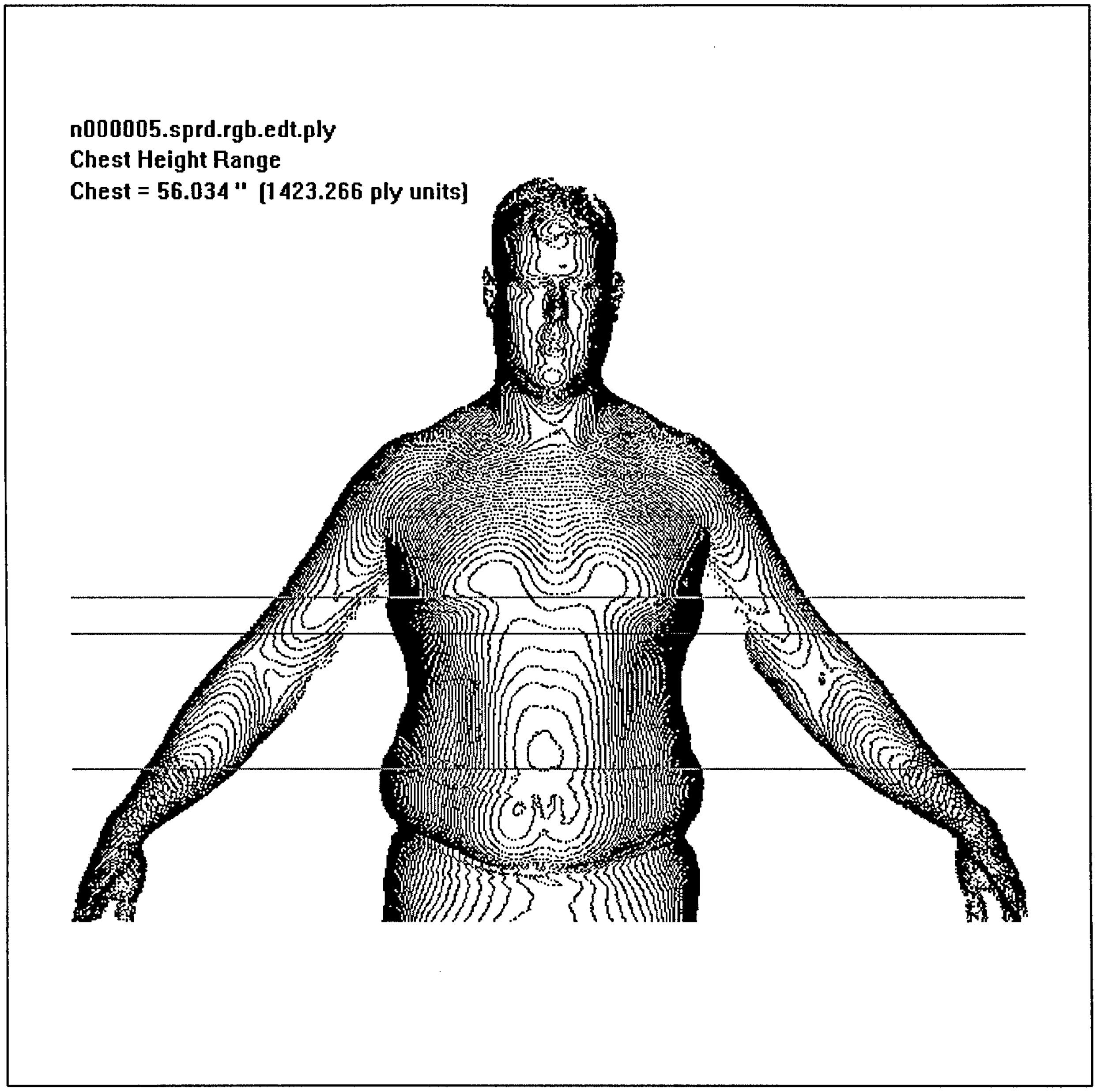

n000004.sprd.rgb.edt.ply
Seat Height Range

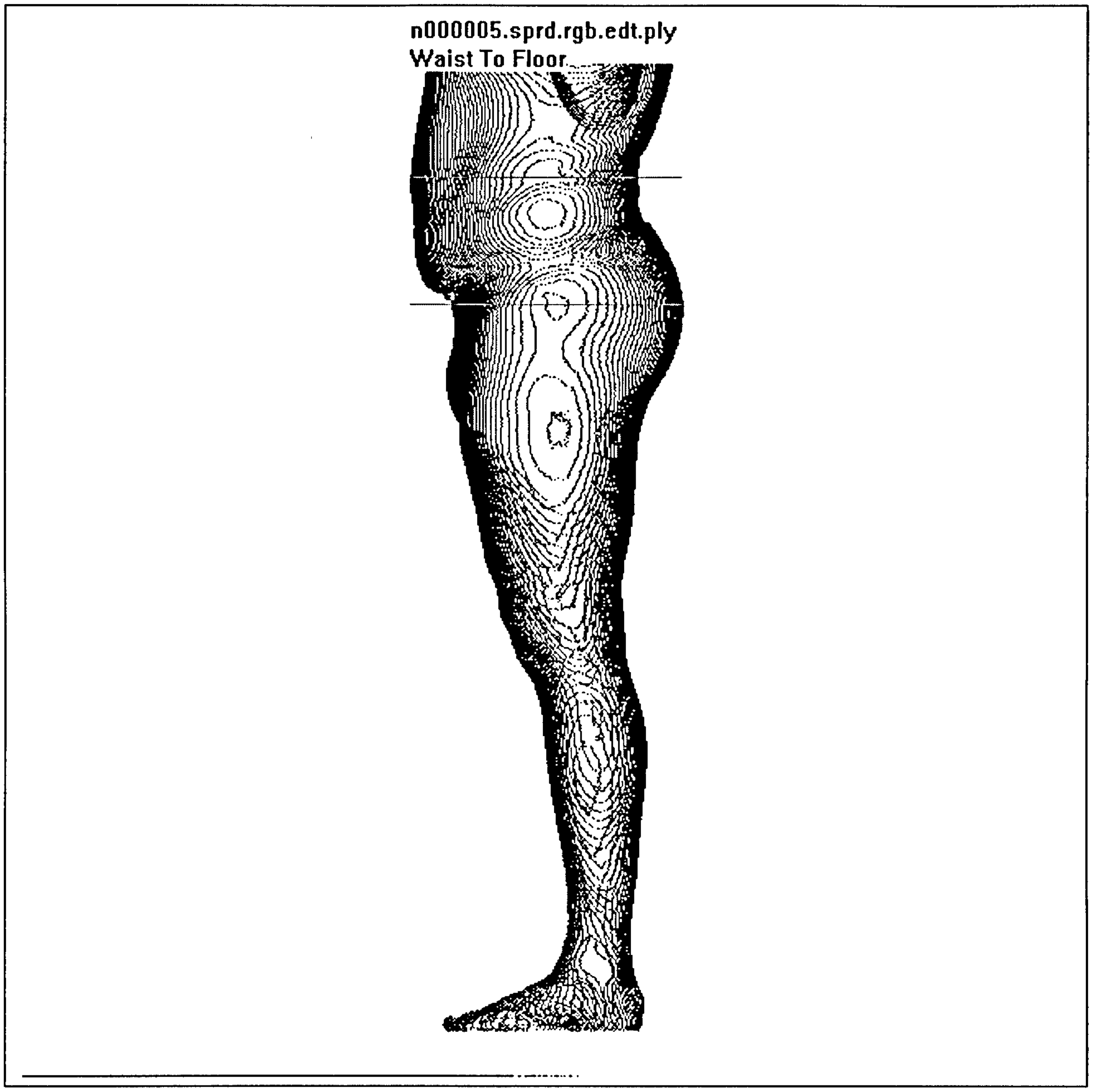

n000004.sprd.rgb.edt.ply
Neck Height Range
Point To Point Line

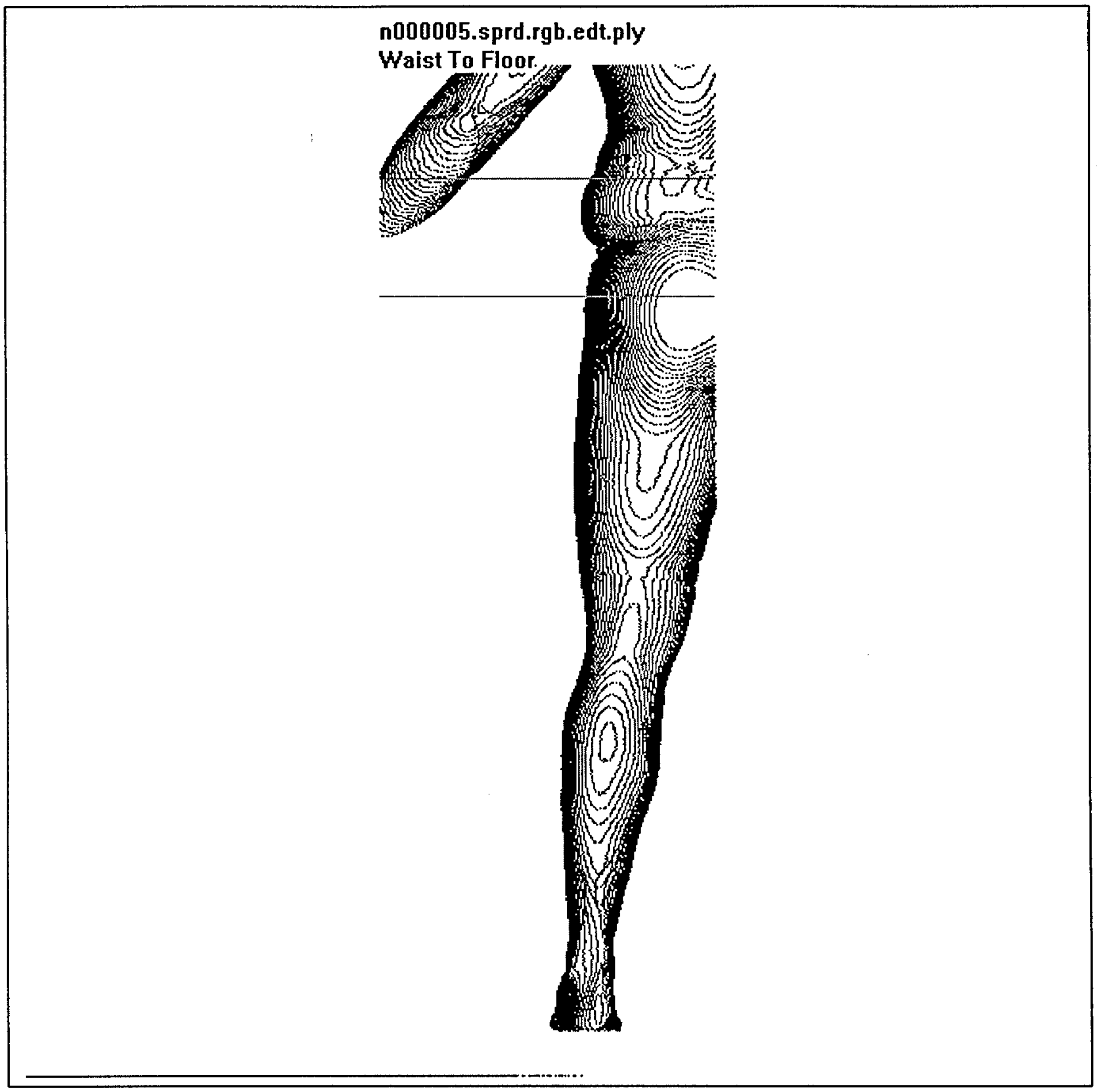

n000004.sprd.rgb.edt.ply

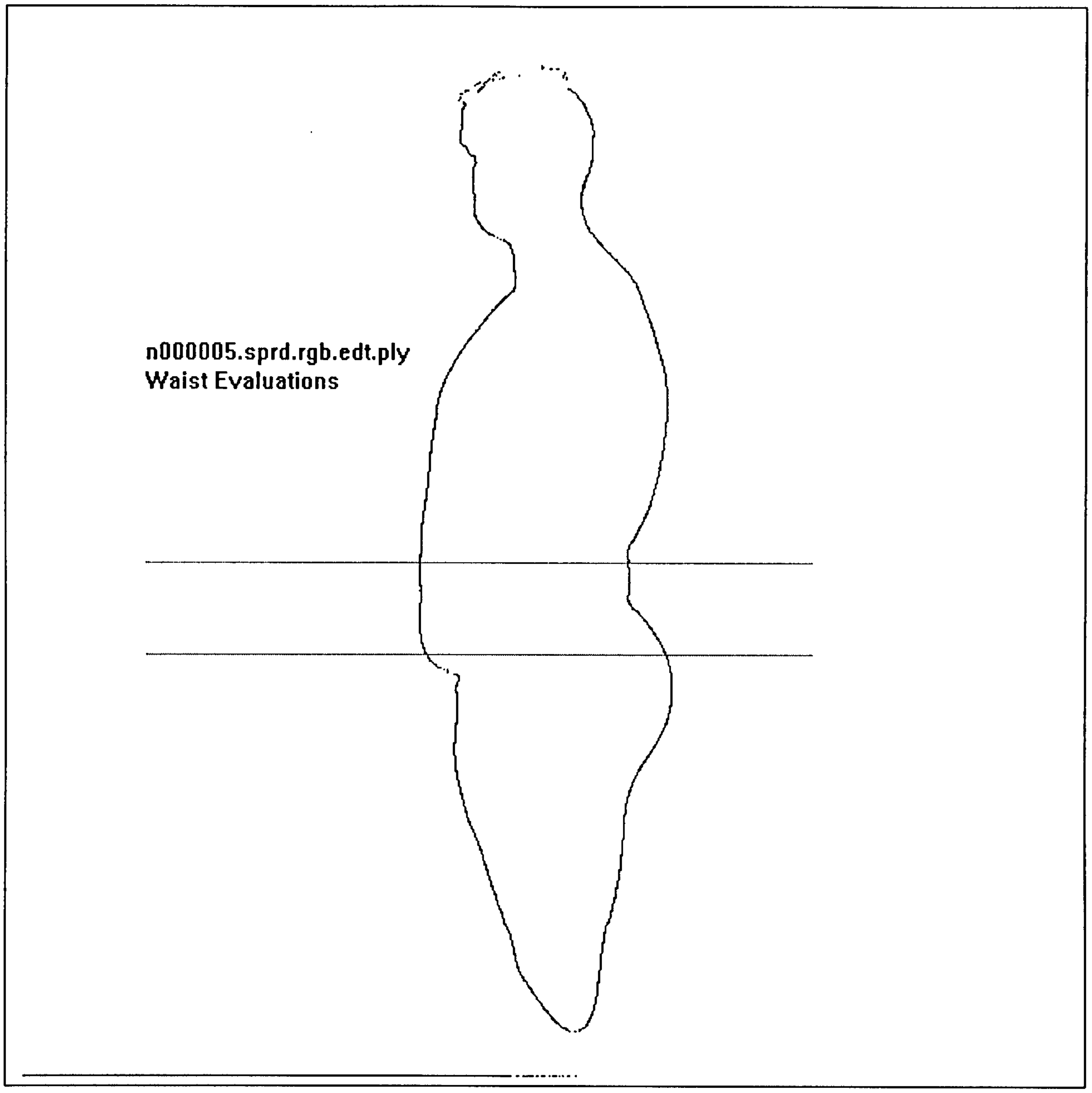

n000004.sprd.rgb.edt.ply
Crotch Height Range

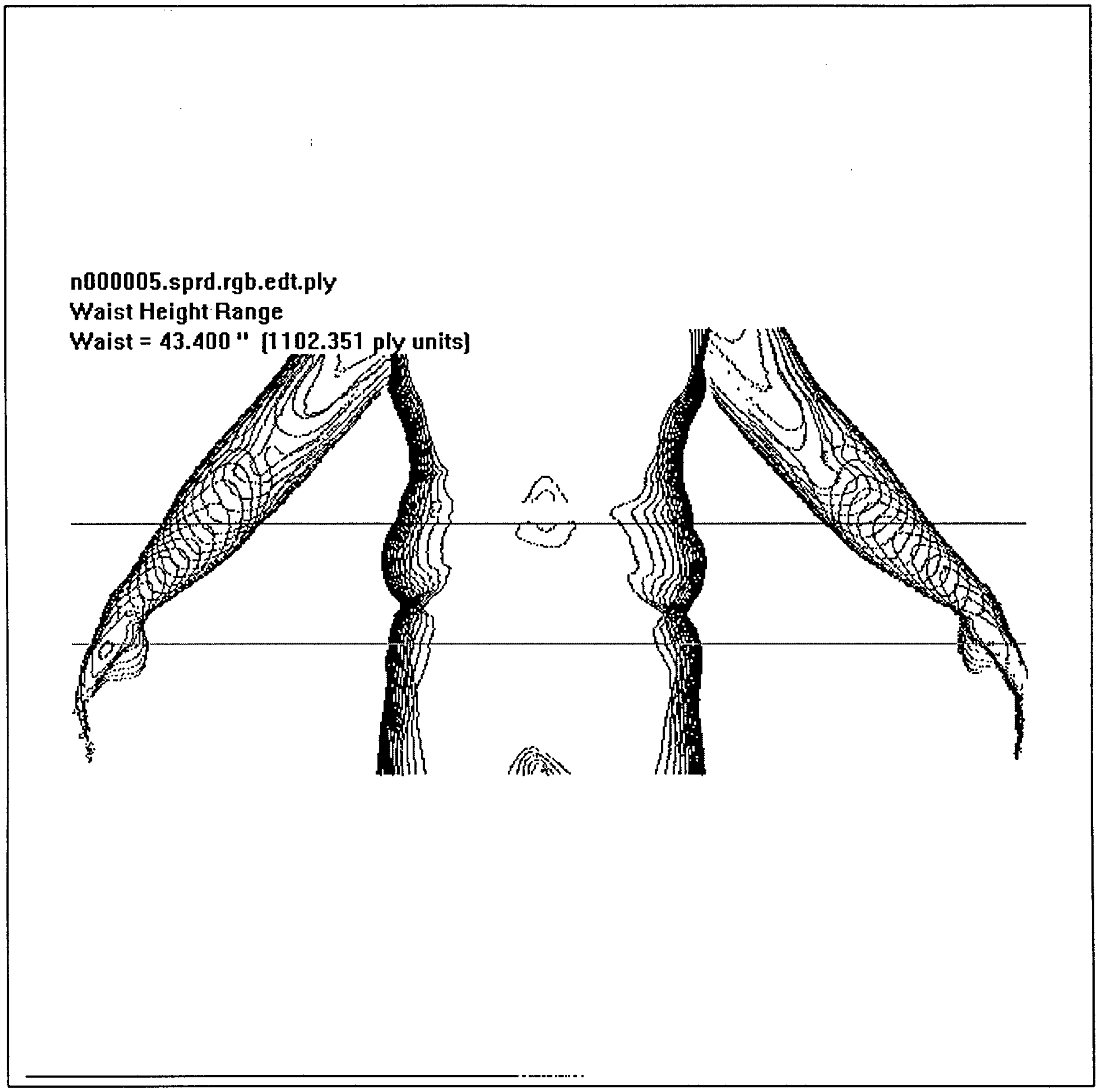

n000004.sprd.rgb.edt.ply
Waist To Floor


n000005.sprd.rgb.edt.ply
Chest Height Range
Chest = 56.034 " (1423.266 ply units)

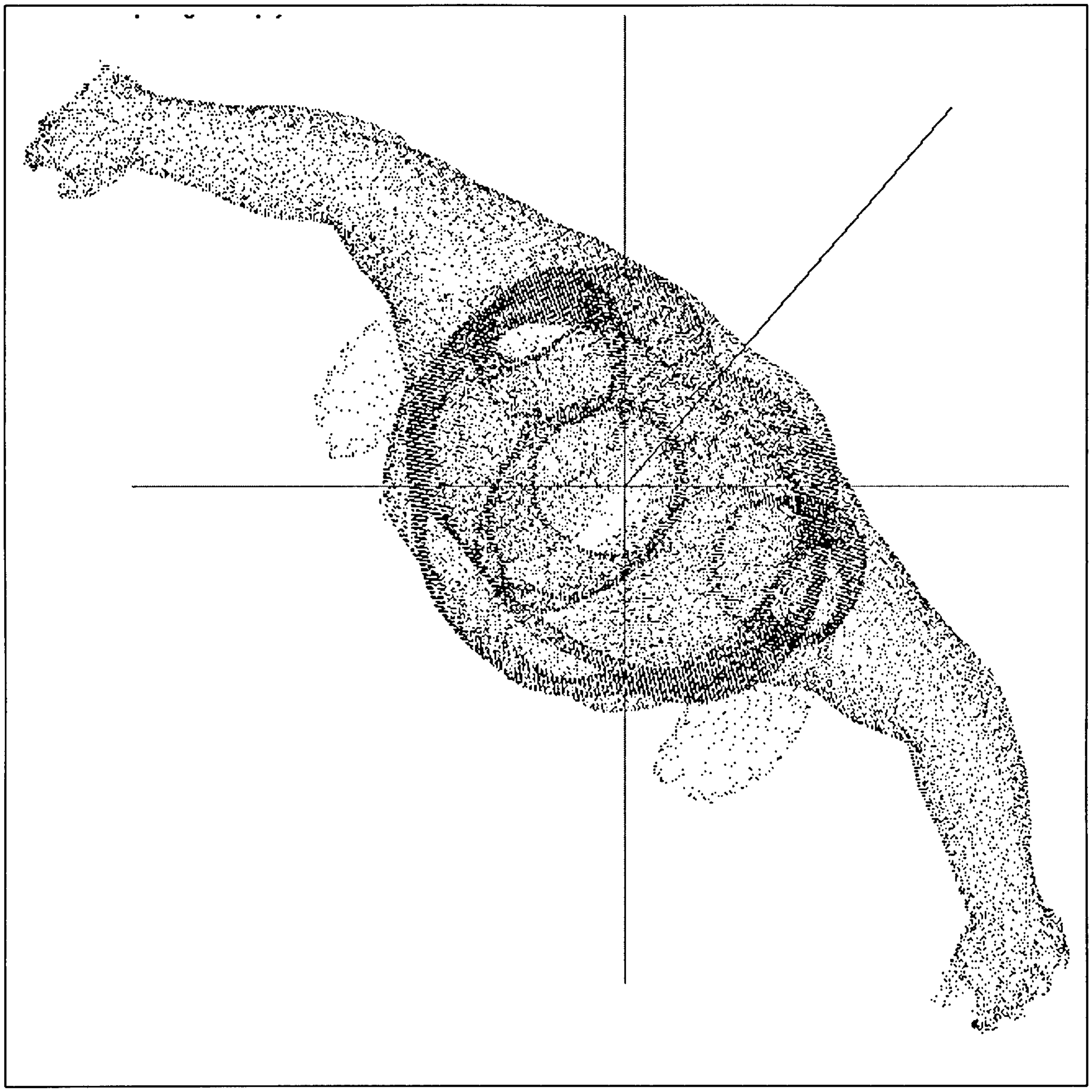

n000005.sprd.rgb.edt.ply
Waist To Floor

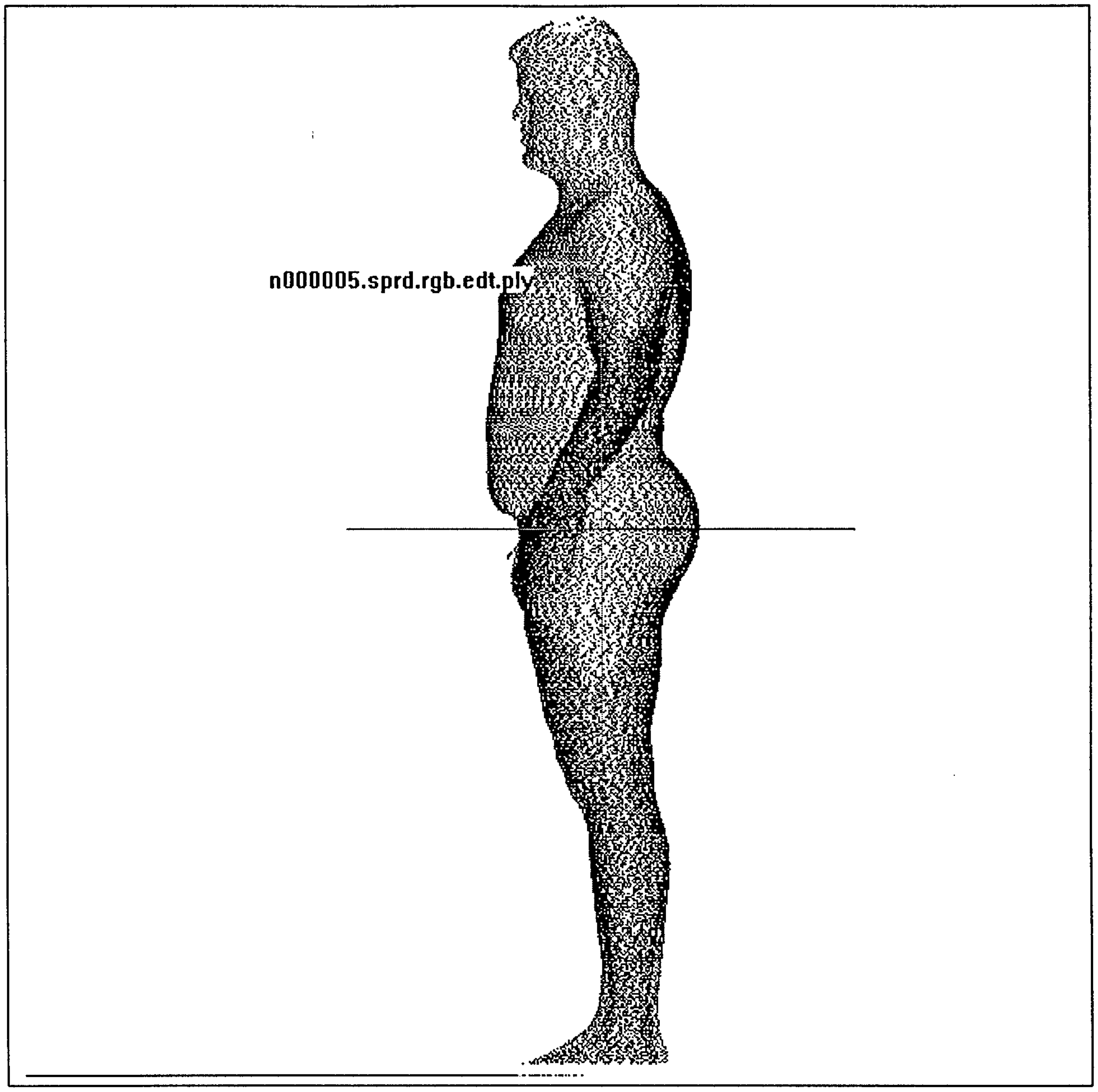

n000005.sprd.rgb.edt.ply
Waist To Floor

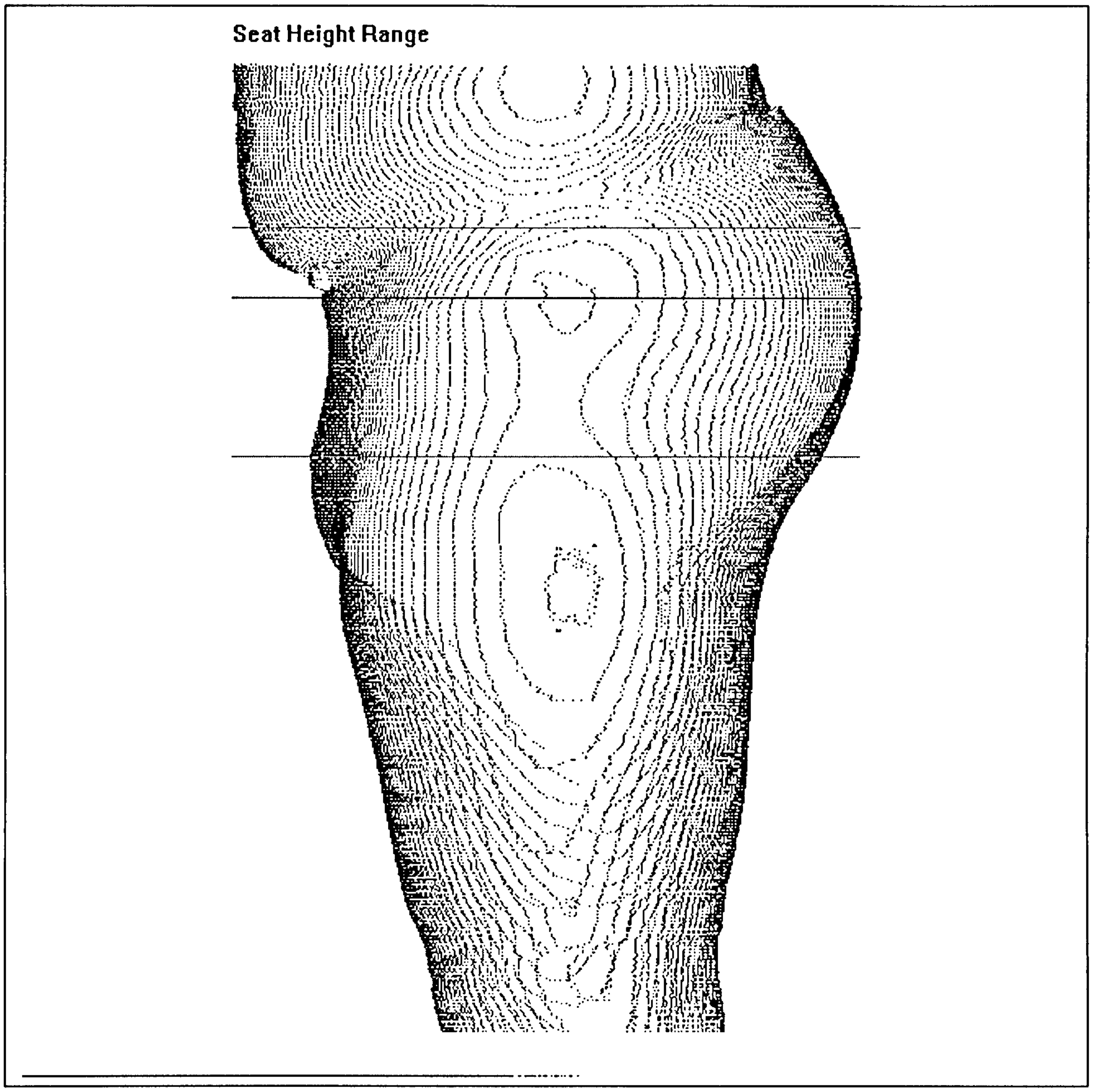

n000005.sprd.rgb.edt.ply
Waist Evaluations

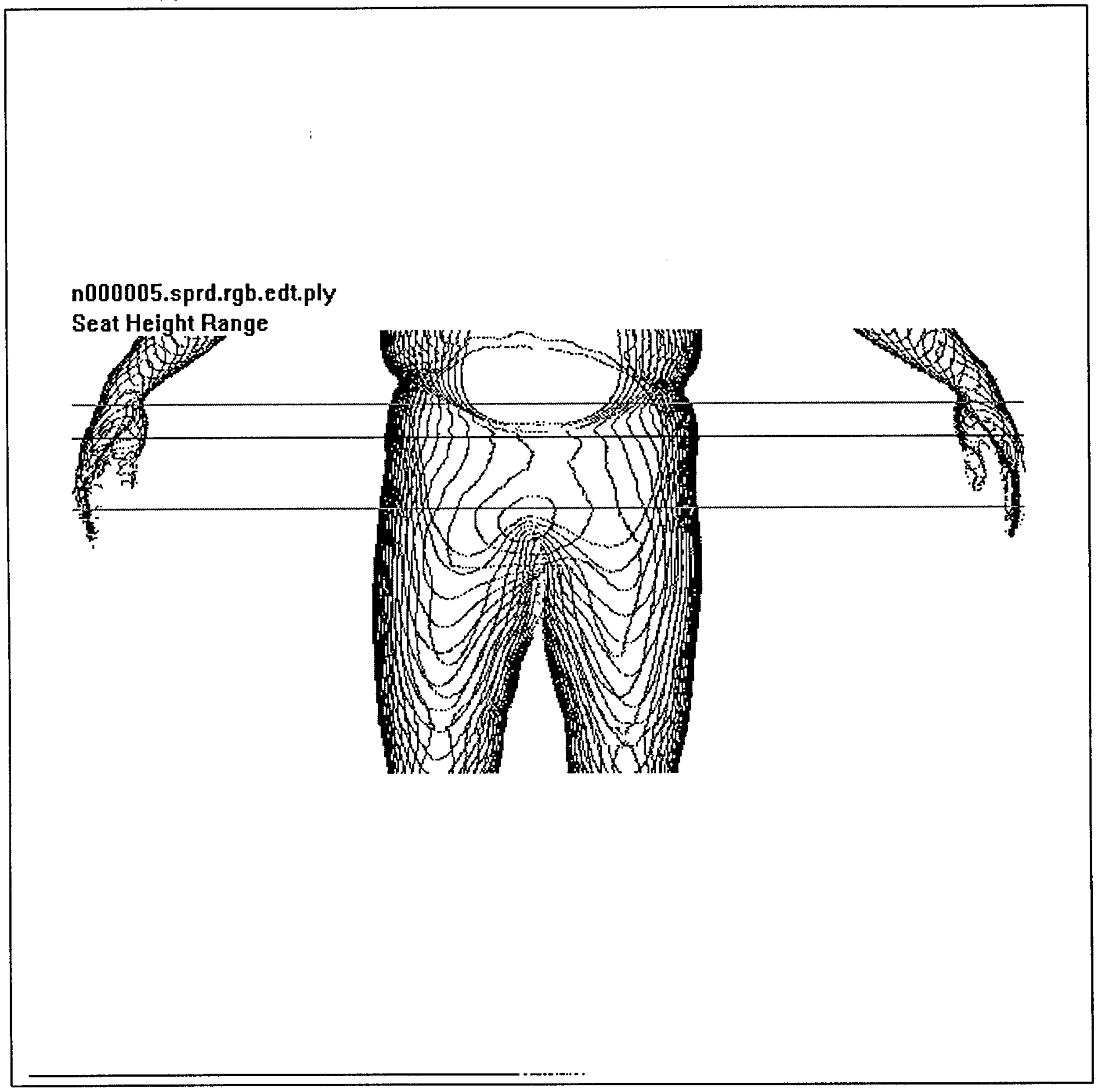


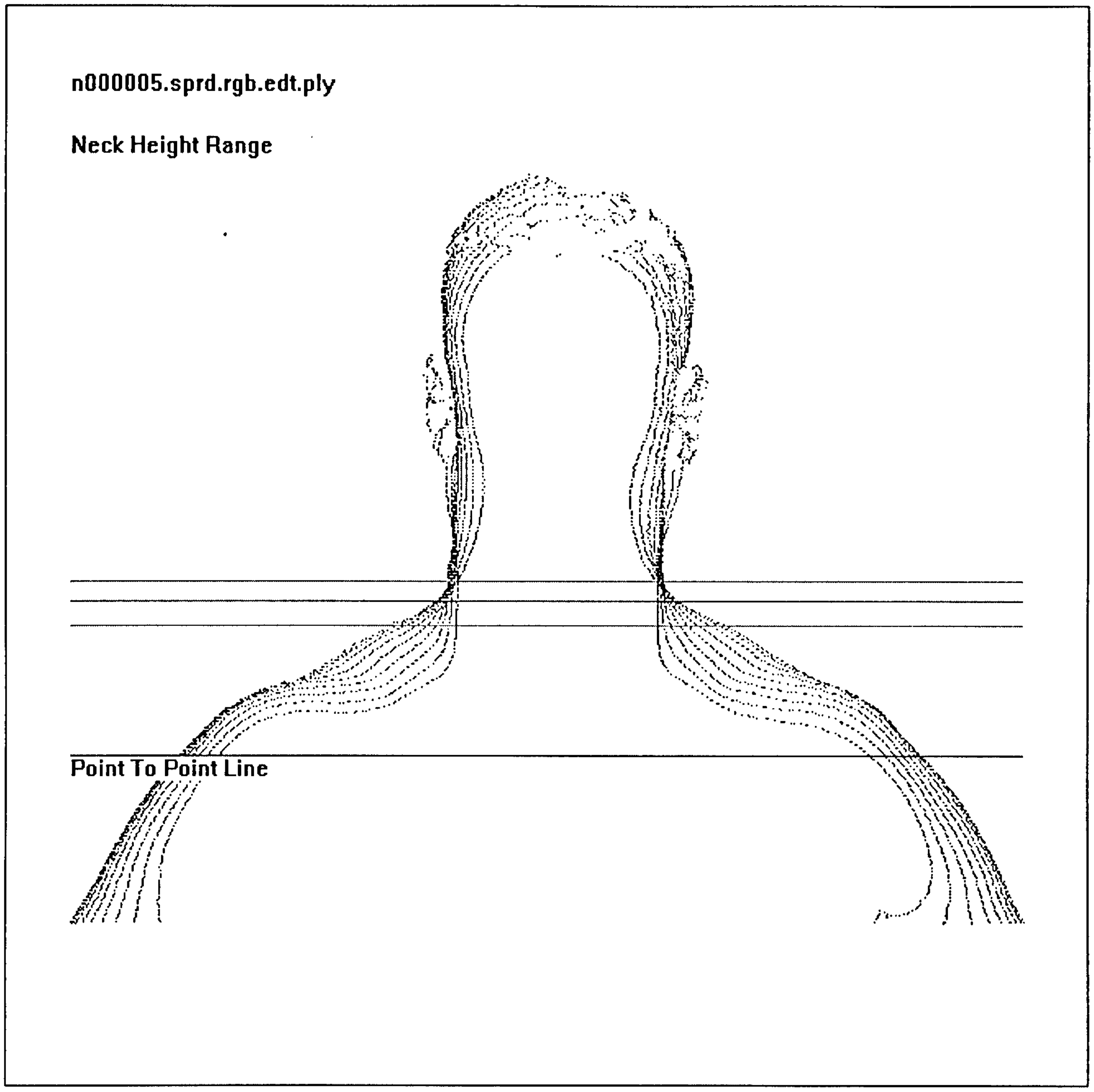
n000005.sprd.rgb.edt.ply
Waist Height Range
Waist = 43.400 " [1102.351 ply units]

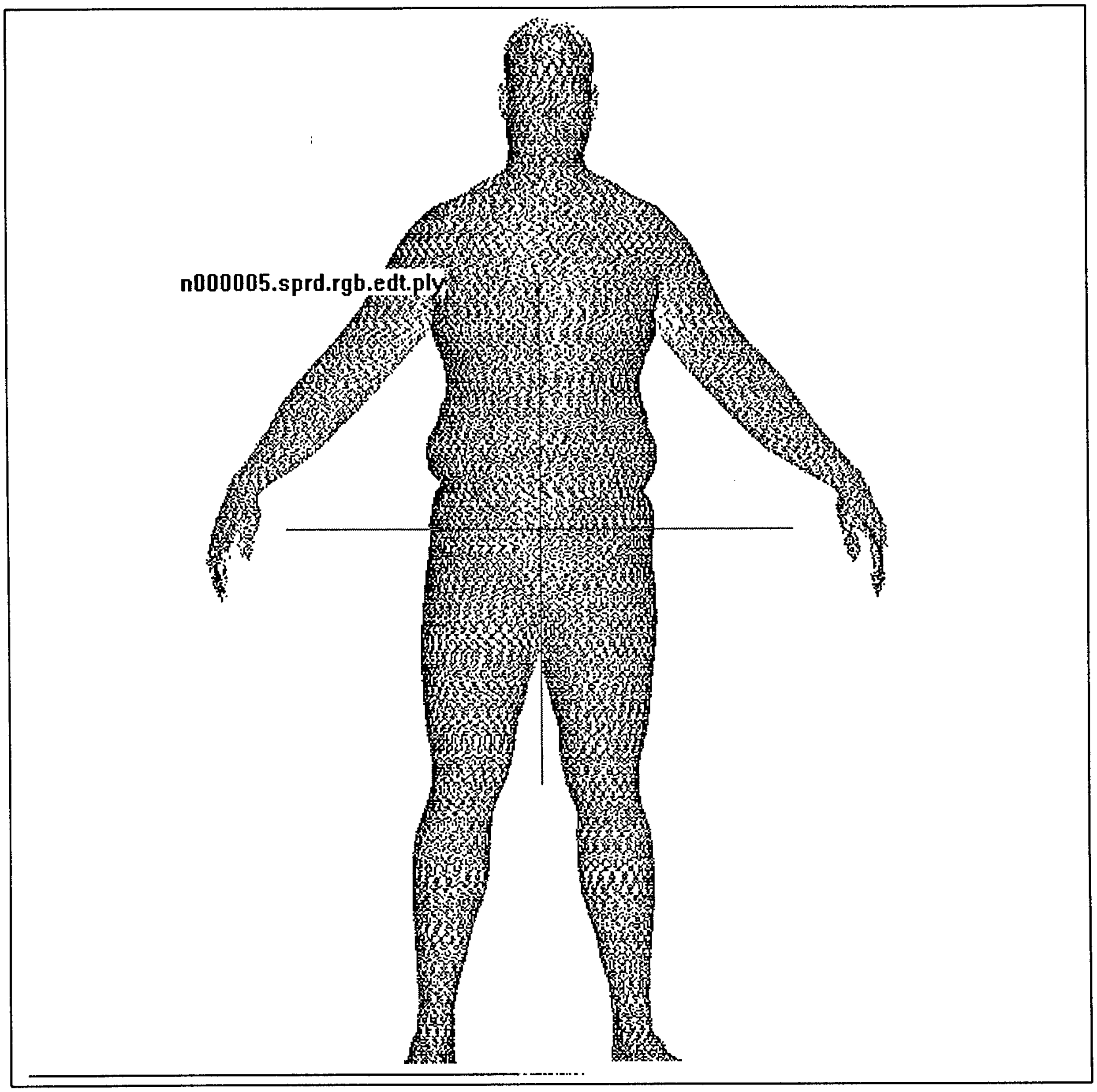


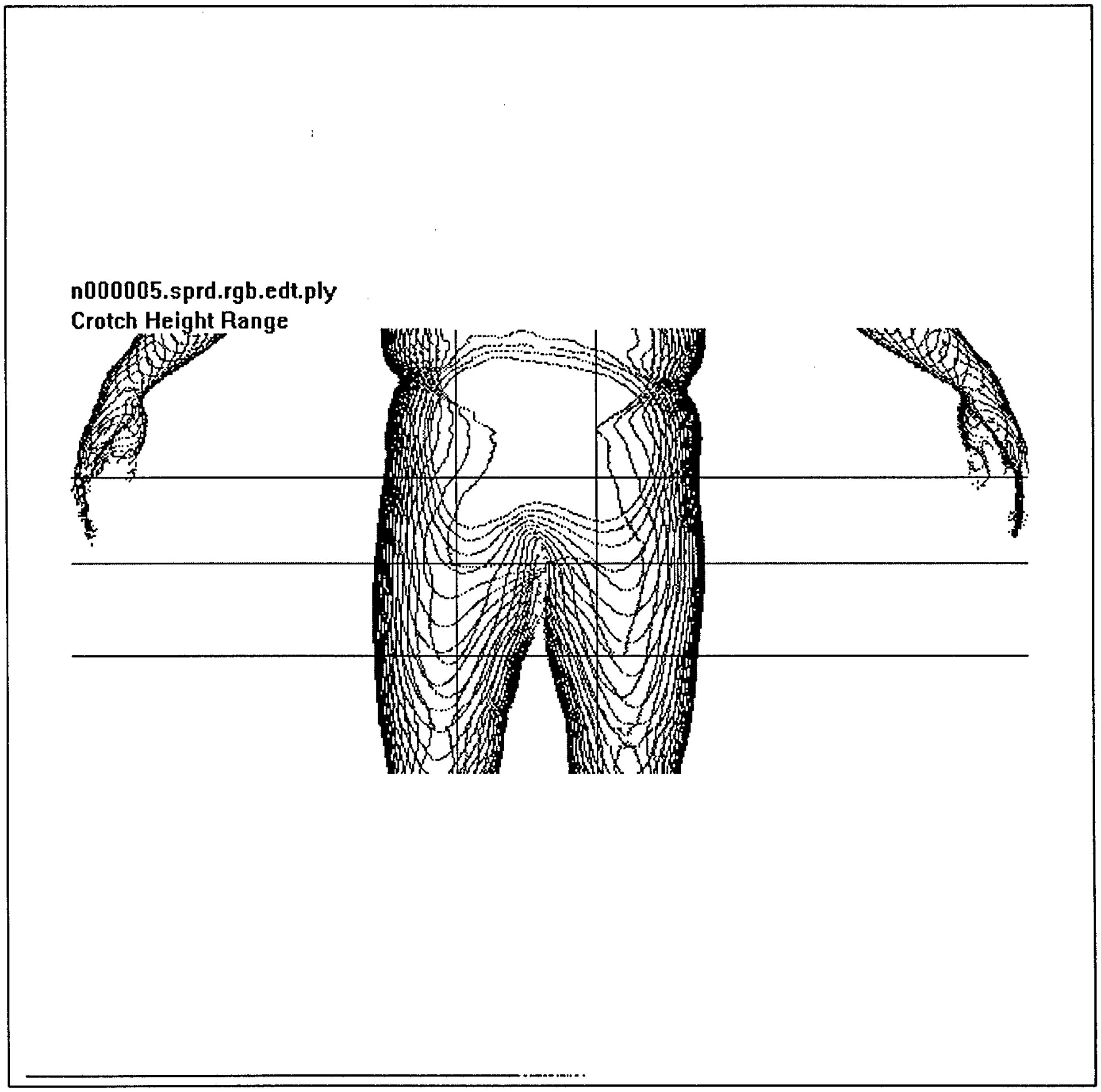

n000005.sprd.rgb.edt.ply

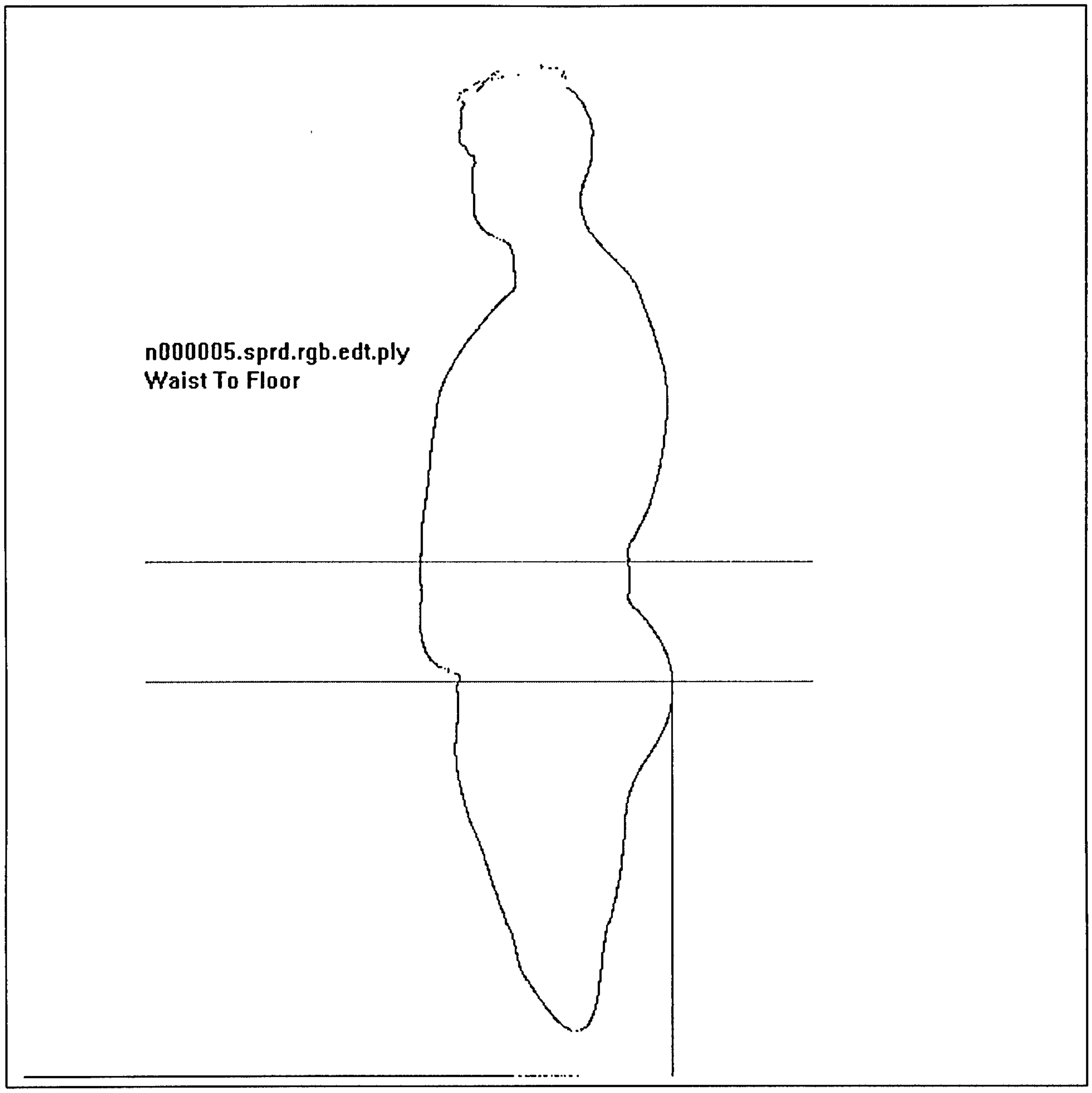

Seat Height Range

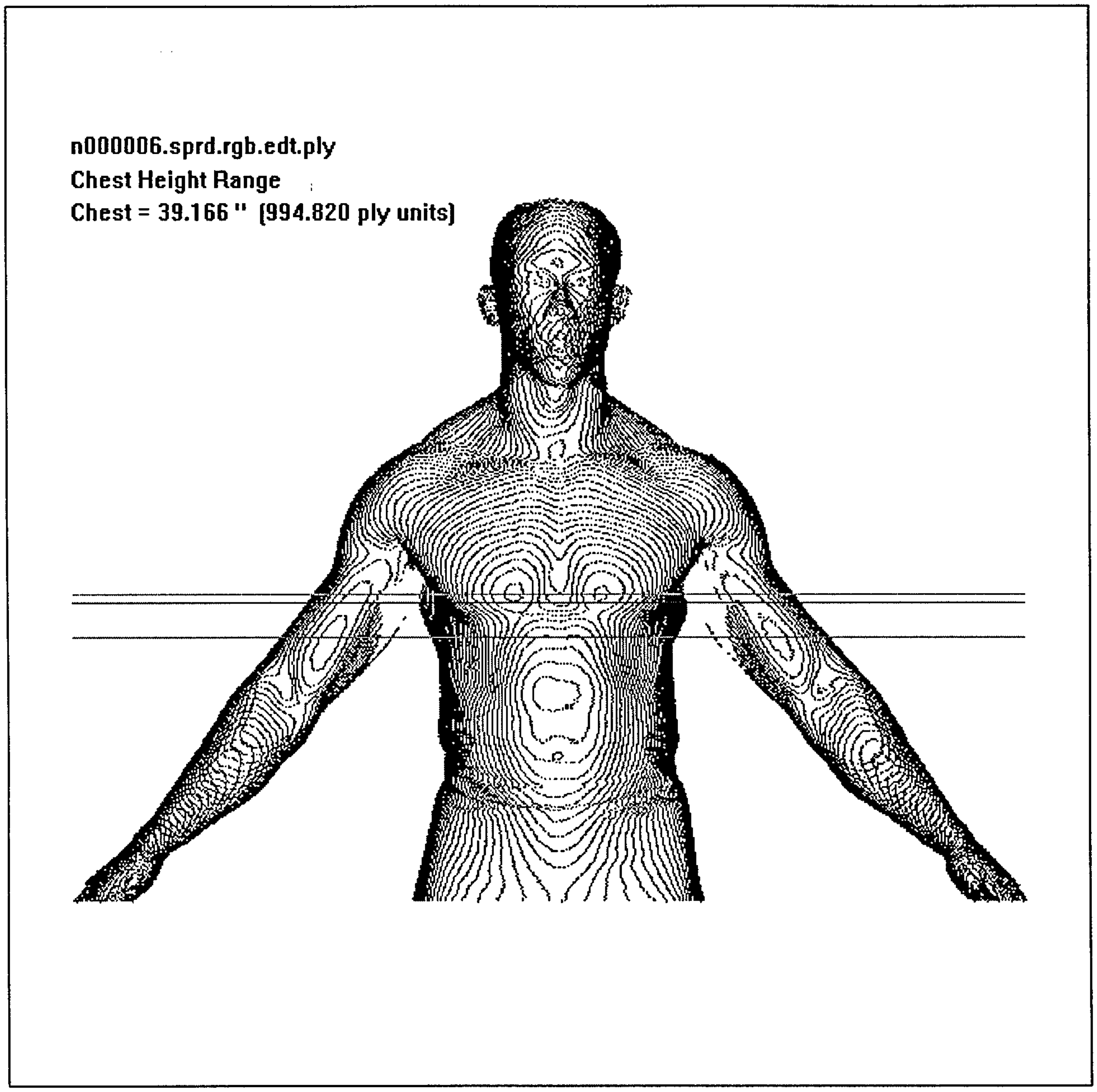

n000005.sprd.rgb.edt.ply
Seat Height Range

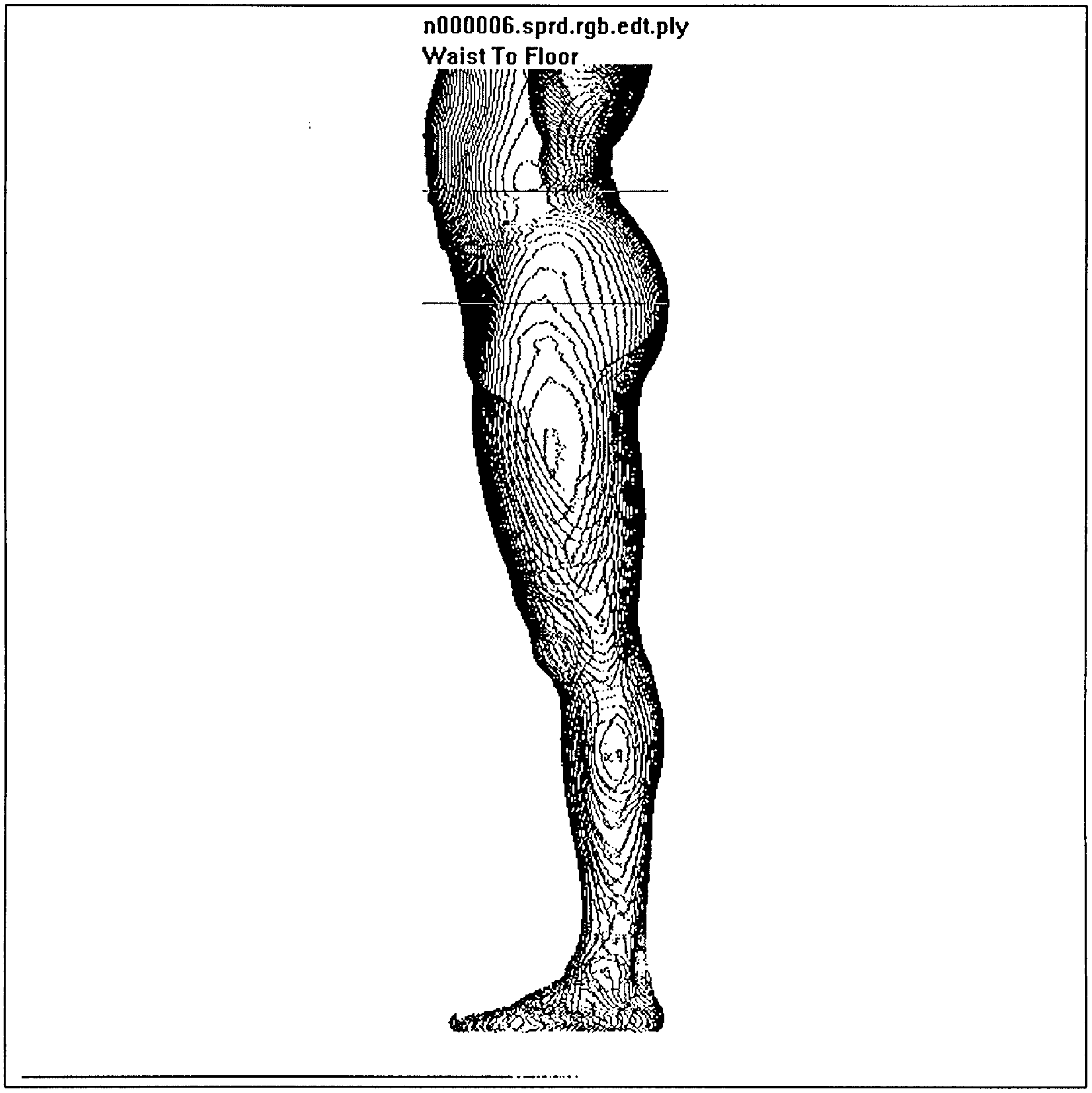

n000005.sprd.rgb.edt.ply
Neck Height Range
Point To Point Line

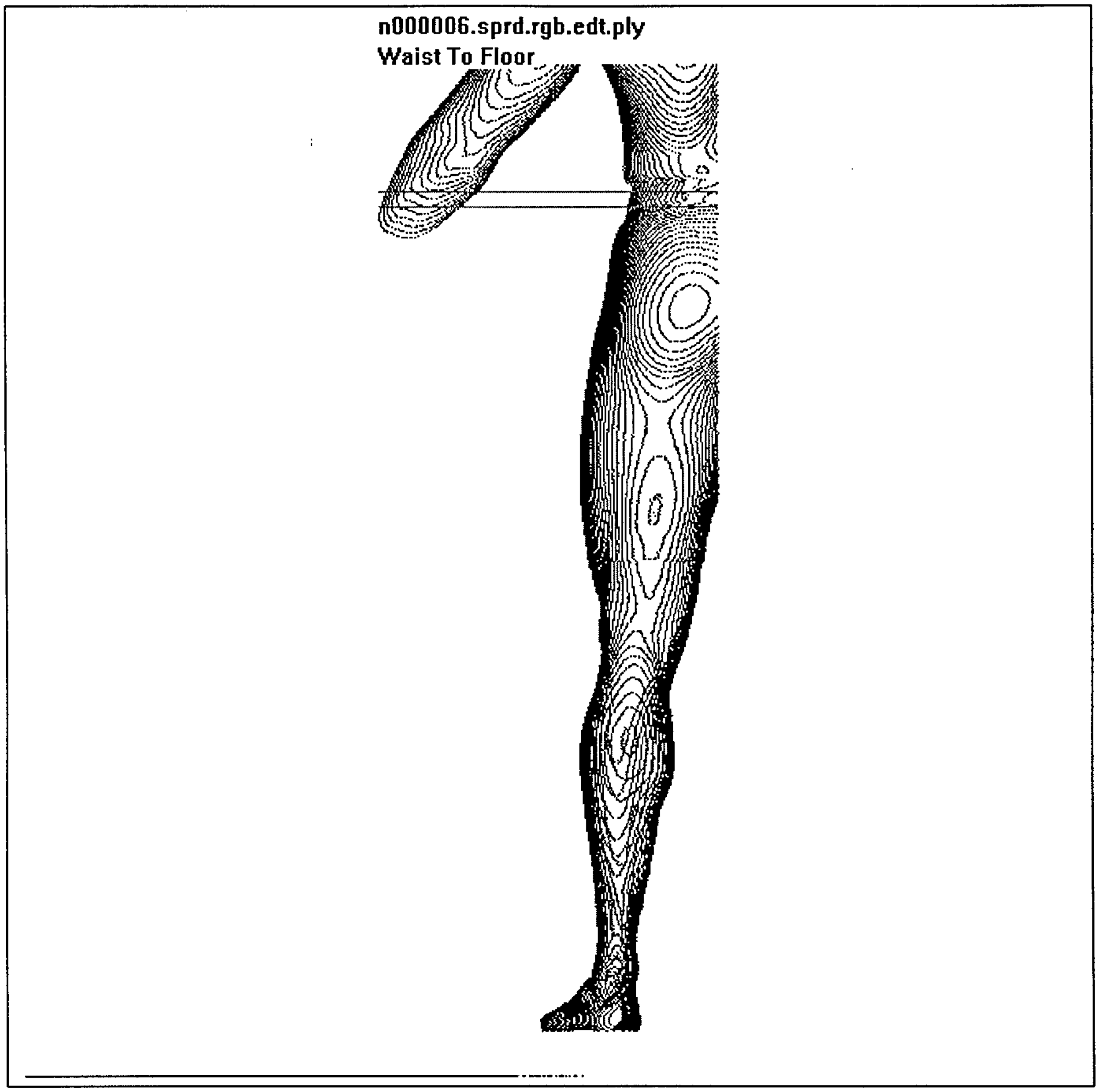

n000005.sprd.rgb.edt.ply

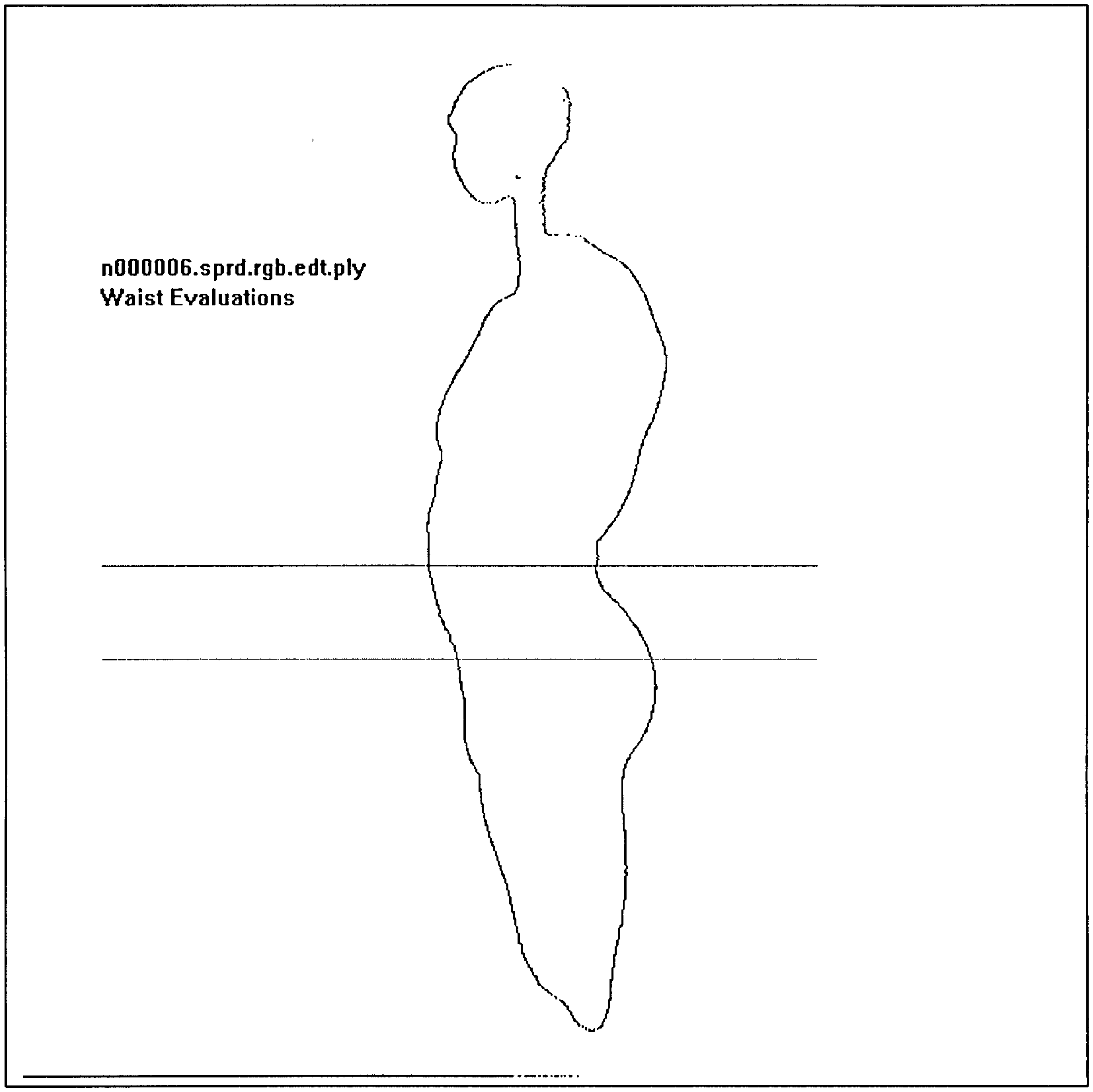

n000005.sprd.rgb.edt.ply
Crotch Height Range

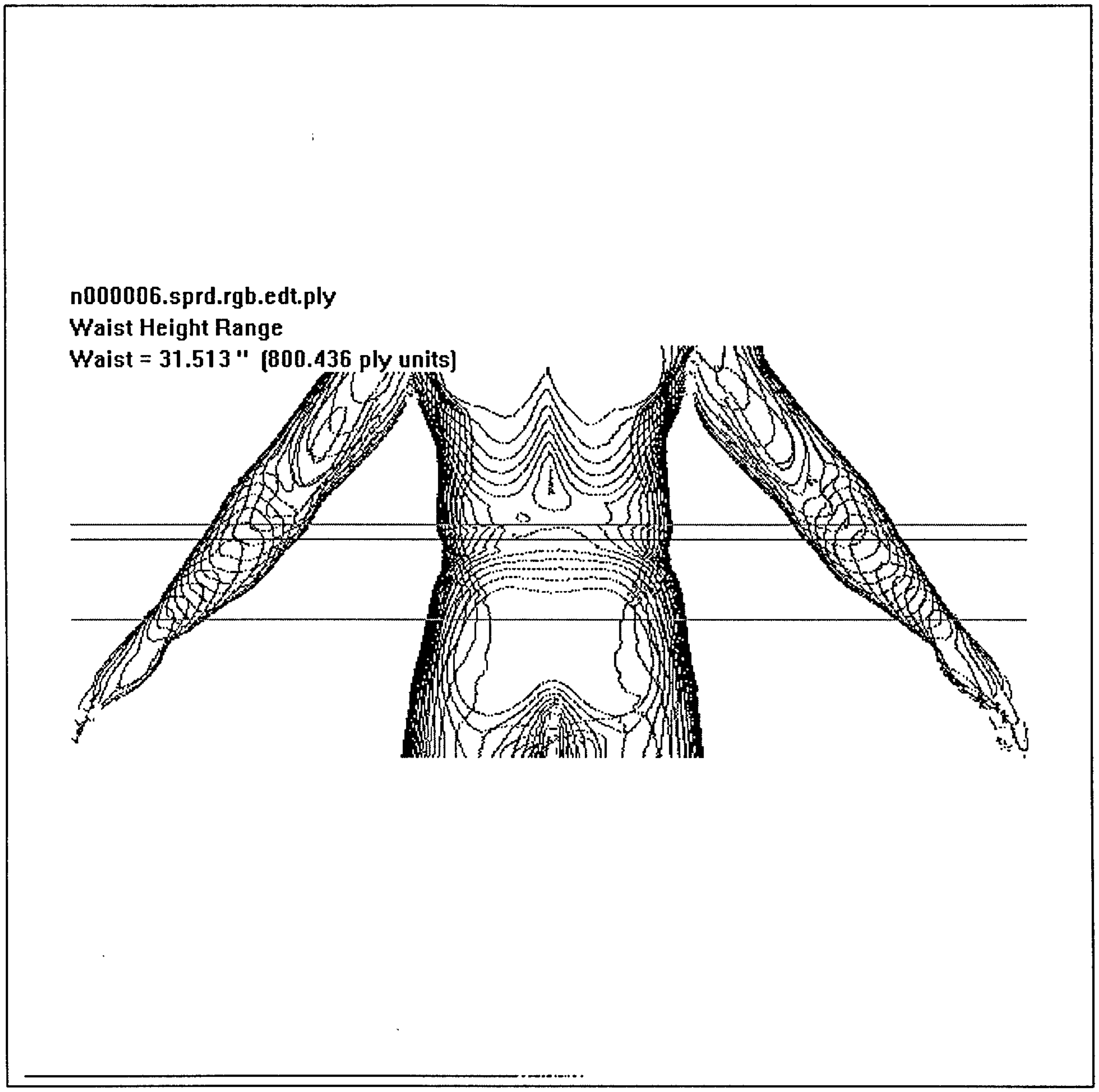

n000005.sprd.rgb.edt.ply
Waist To Floor

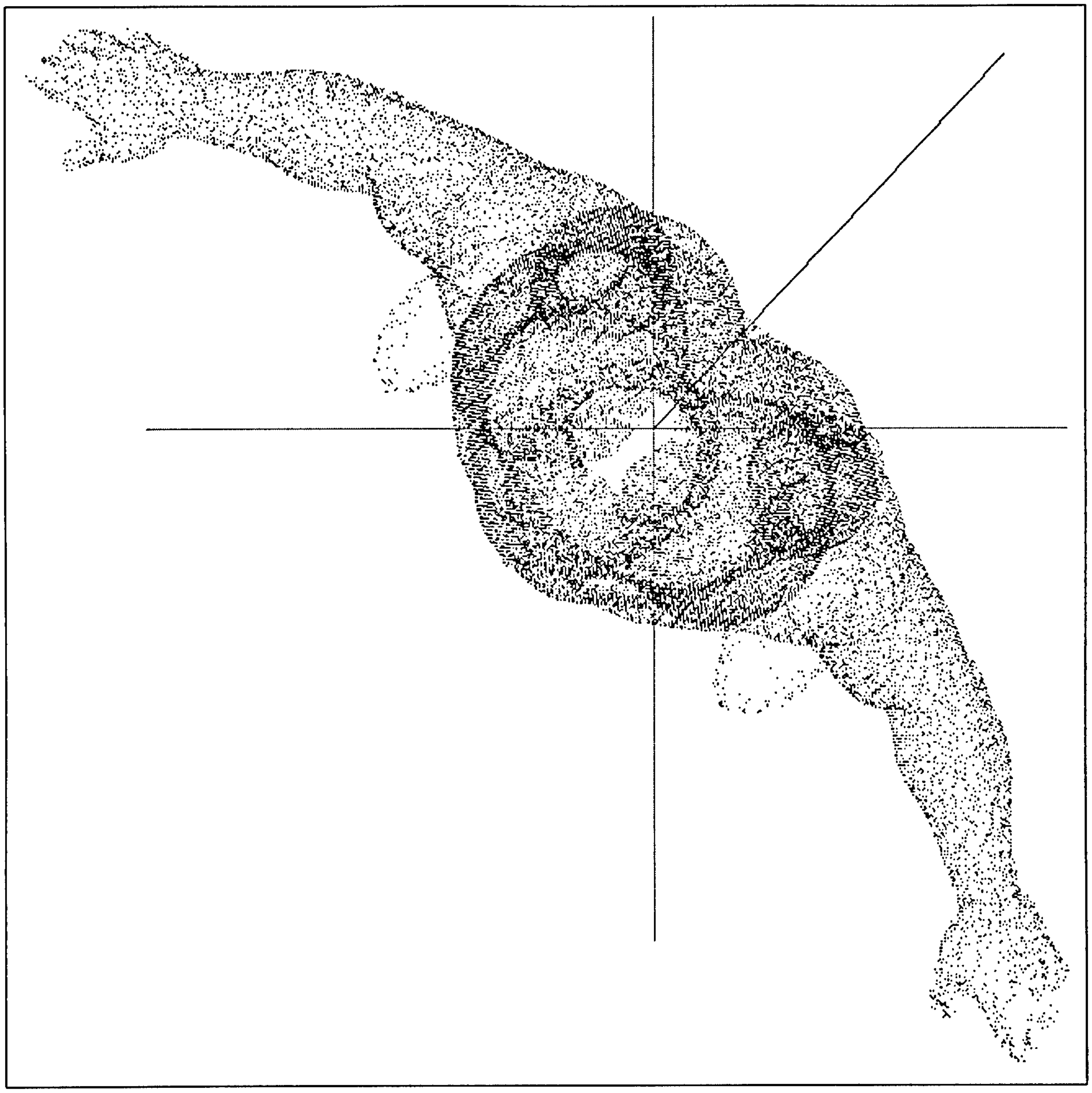

n000006.sprd.rgb.edt.ply
Chest Height Range
Chest = 39.166 " (994.820 ply units)

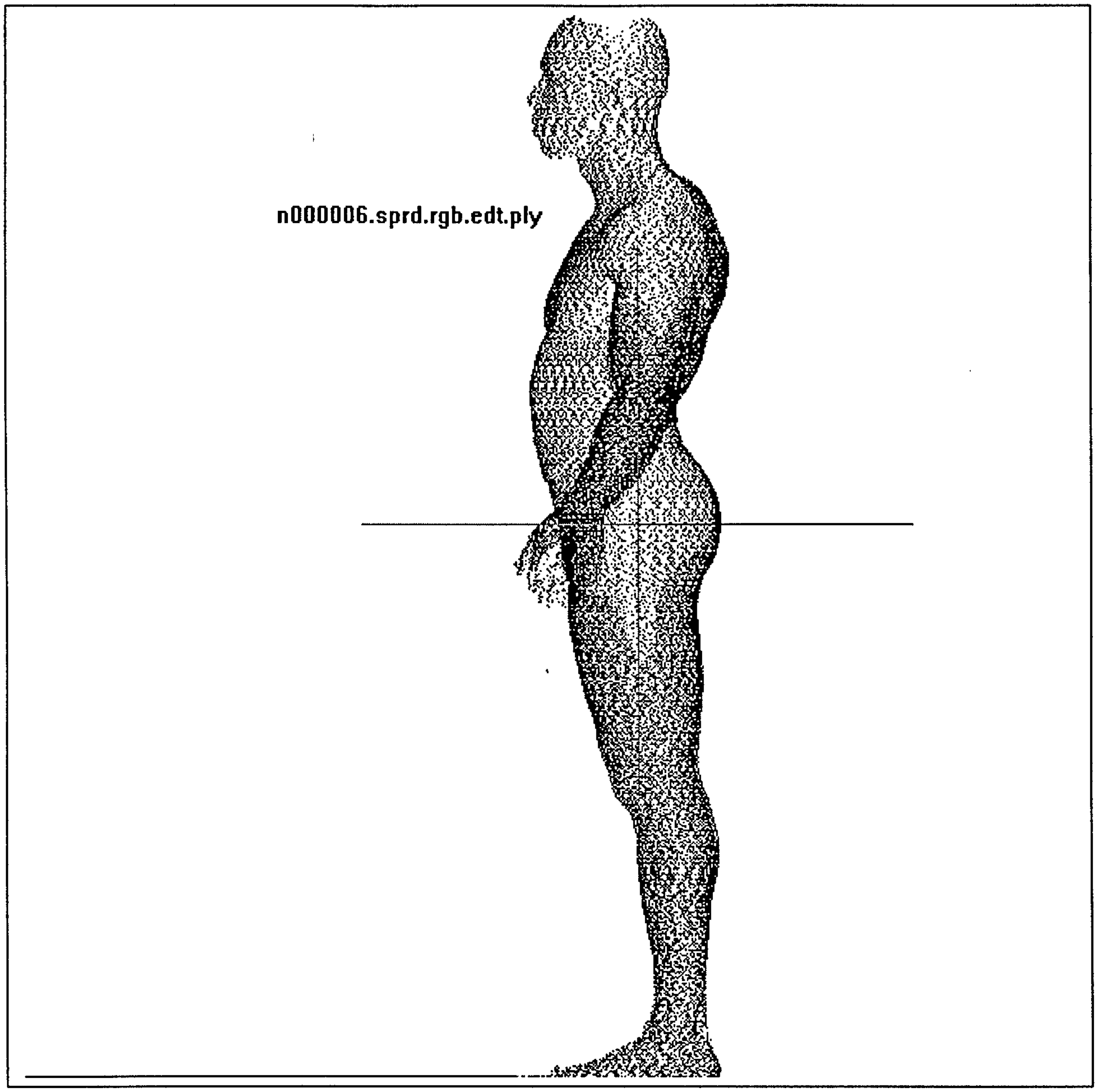

n000006.sprd.rgb.edt.ply
Waist To Floor

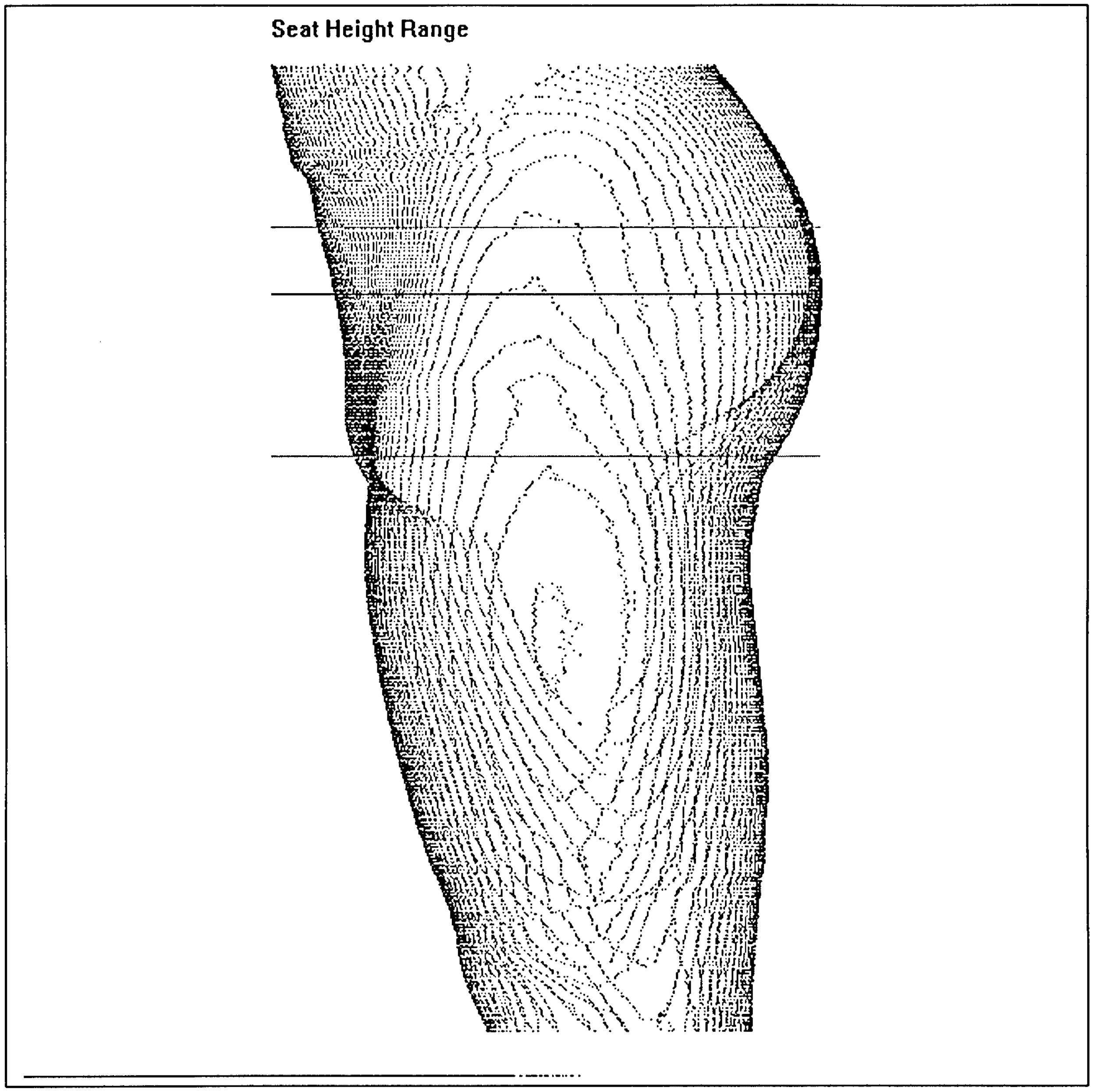

n000006.sprd.rgb.edt.ply
Waist To Floor

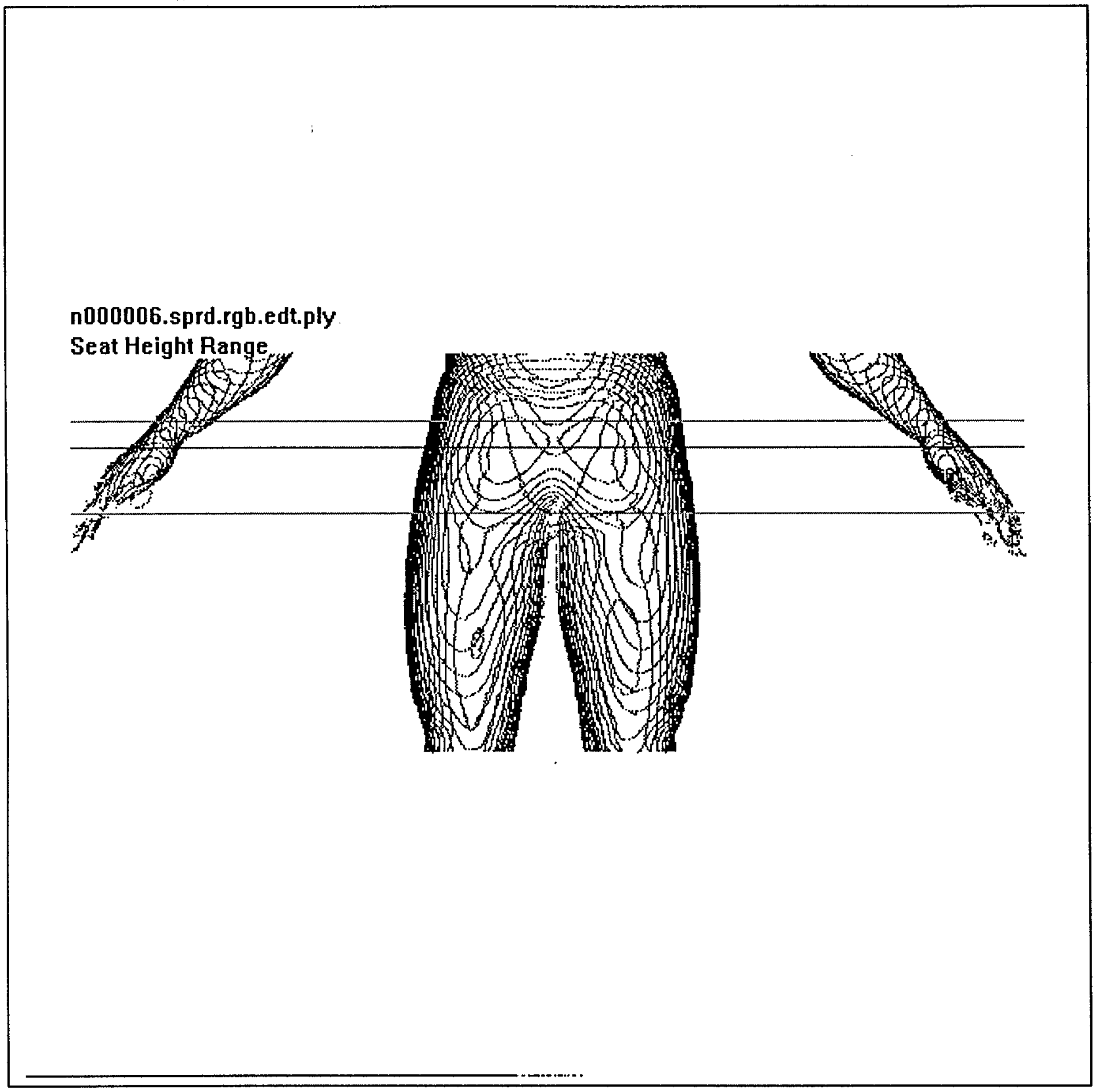

n000006.sprd.rgb.edt.ply
Waist Evaluations

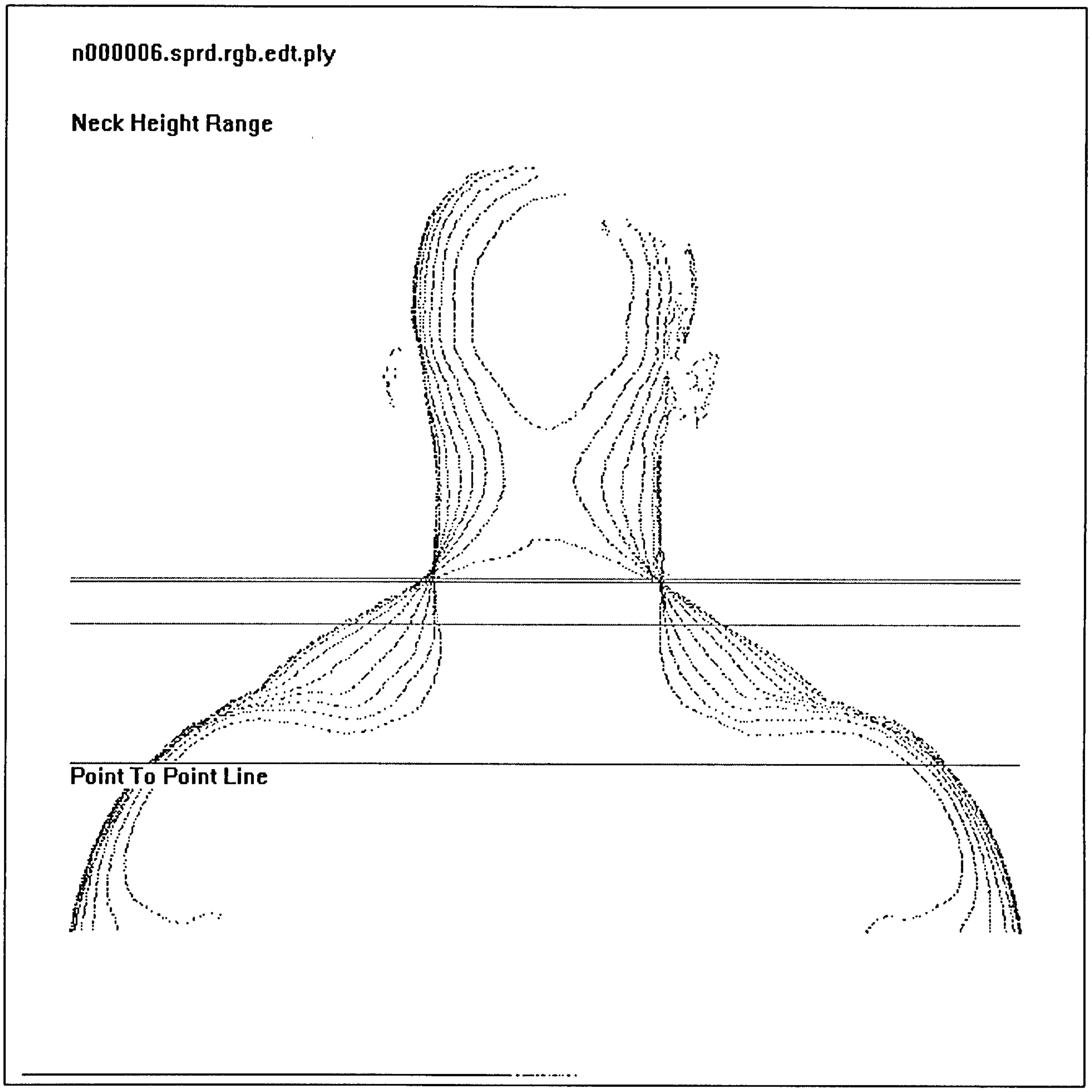

n000006.sprd.rgb.edt.ply
Waist Height Range
Waist = 31.513 " (800.436 ply units)

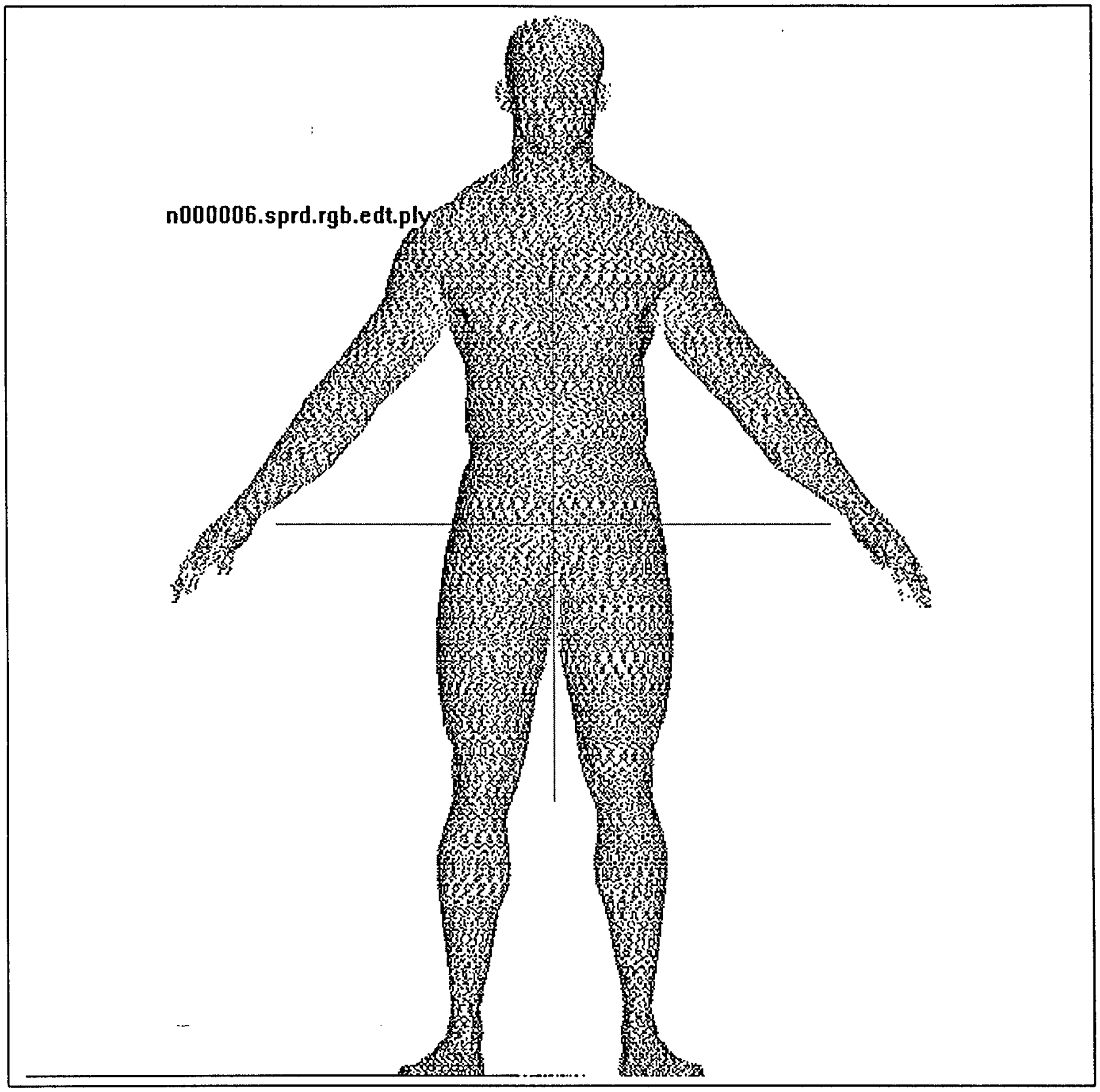


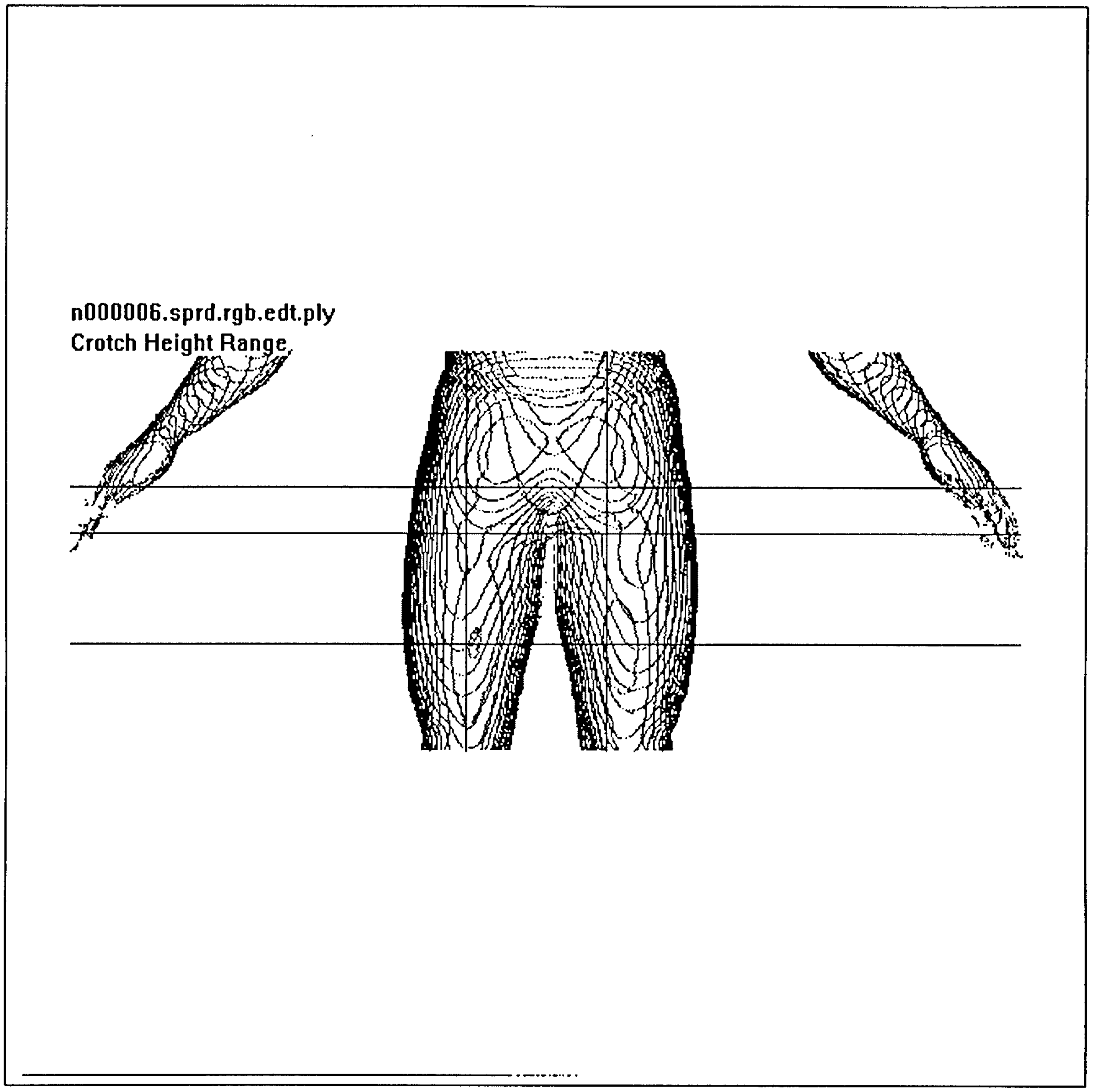

n000006.sprd.rgb.edt.ply

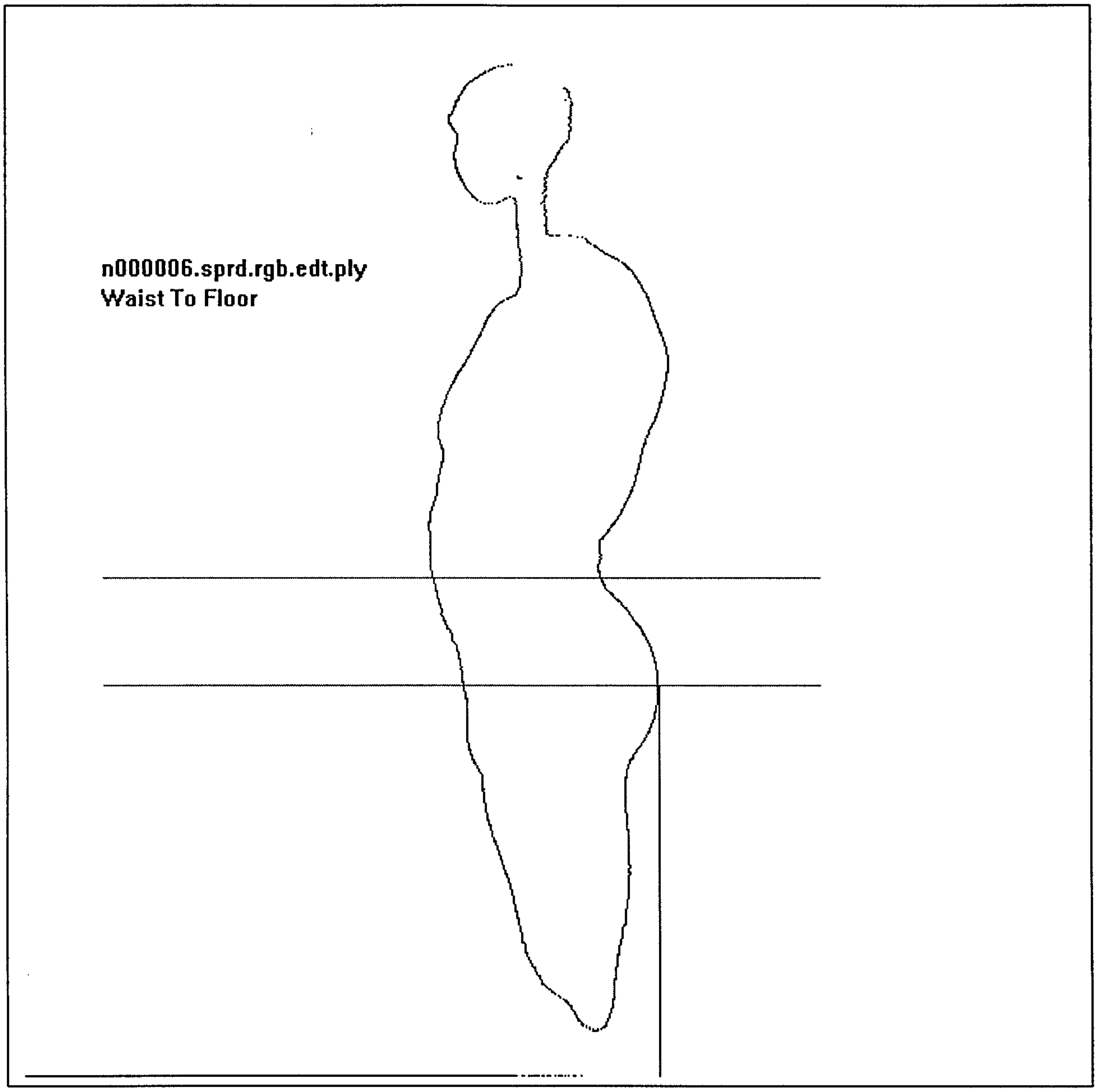

Seat Height Range

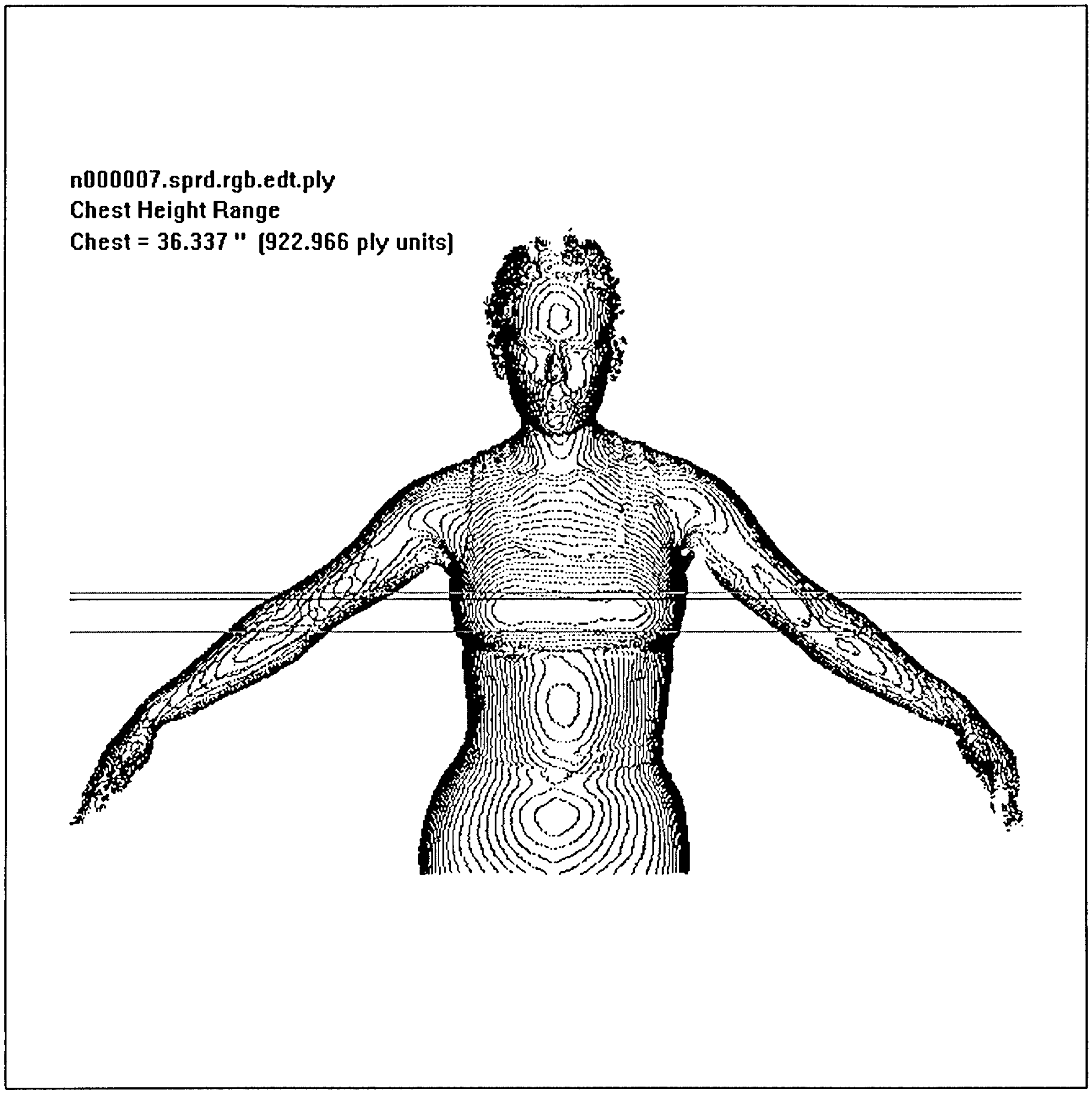

n000006.sprd.rgb.edt.ply
Seat Height Range

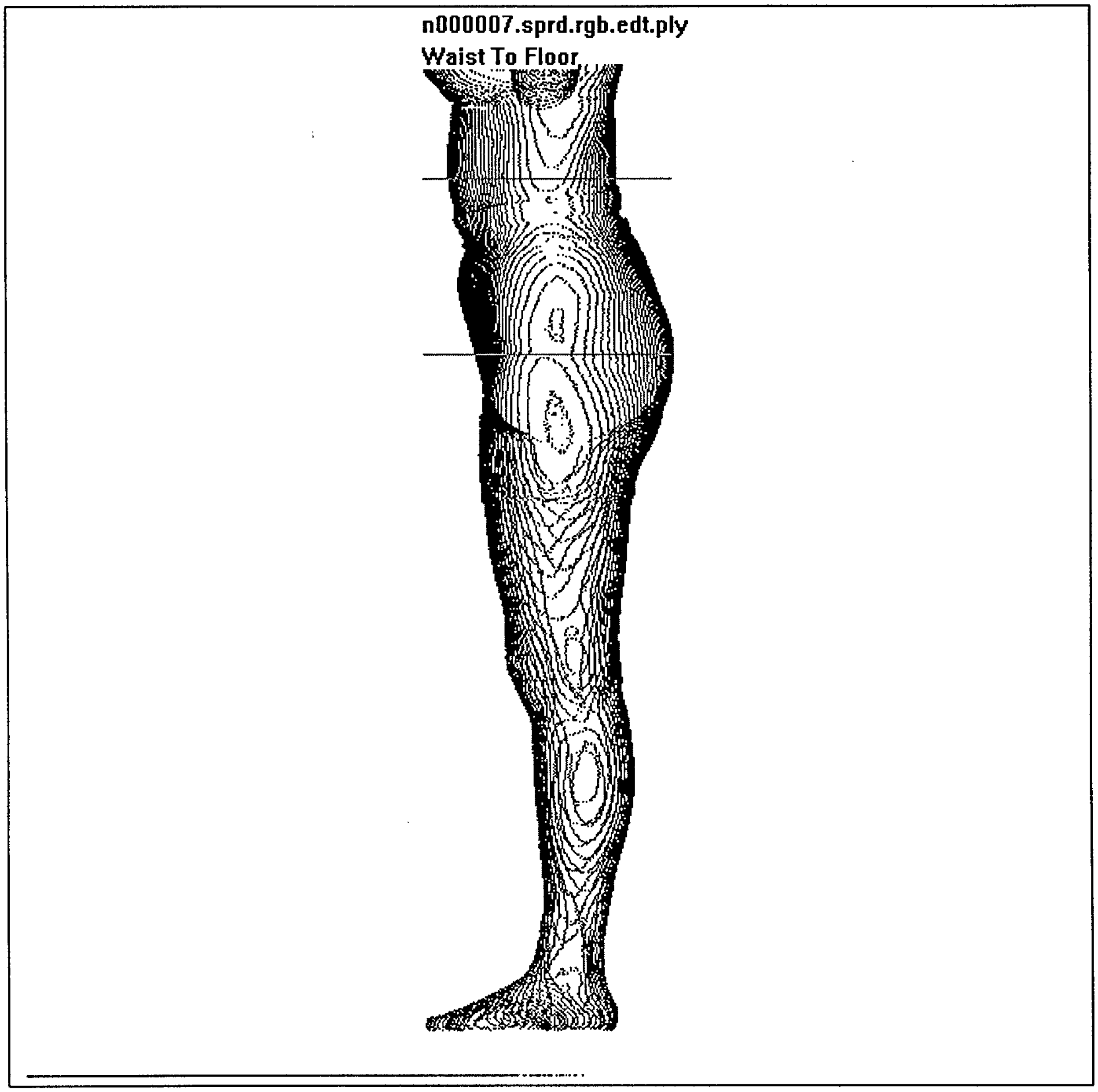

n000006.sprd.rgb.edt.ply
Neck Height Range
Point To Point Line

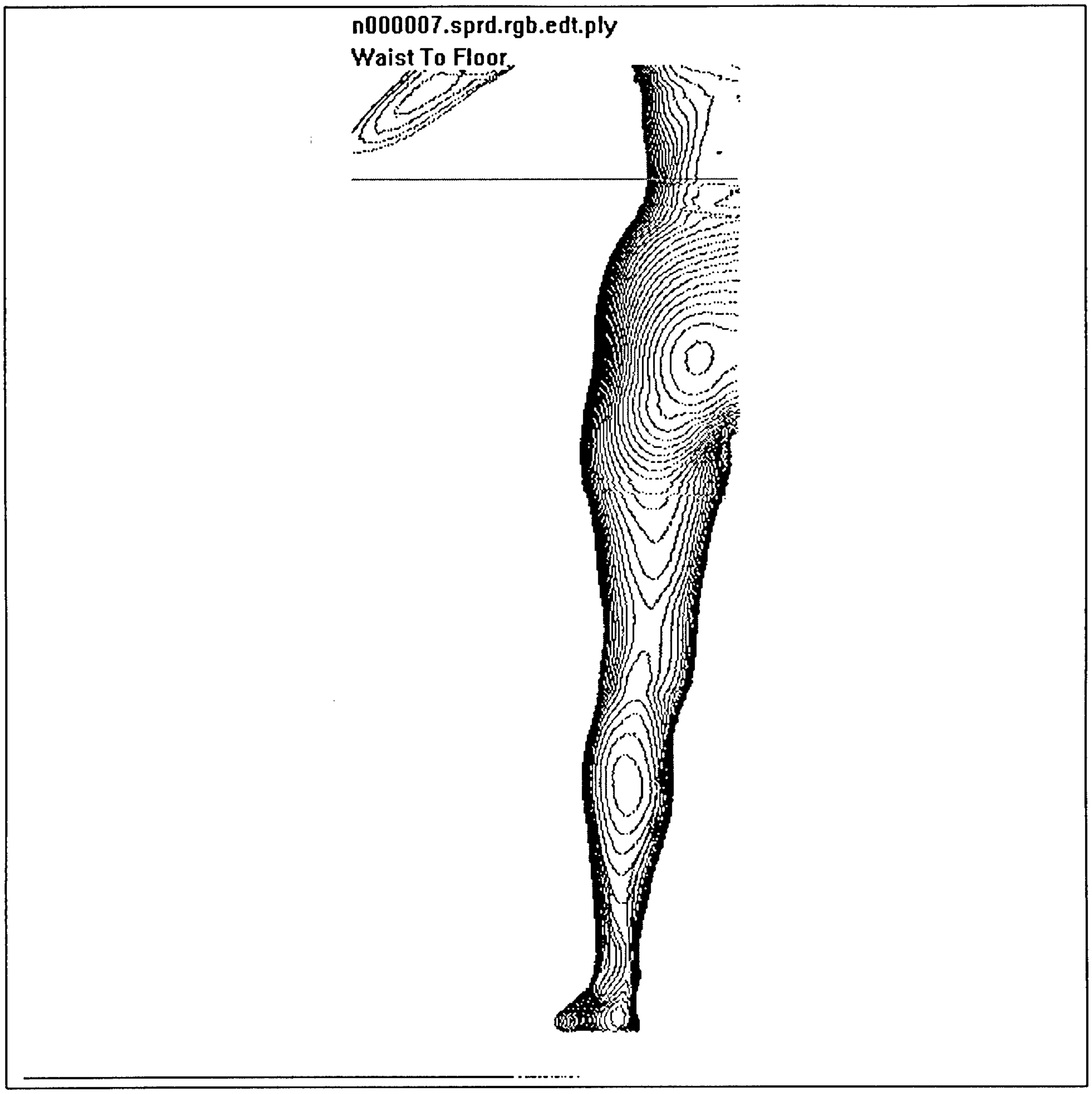

n000006.sprd.rgb.edt.ply

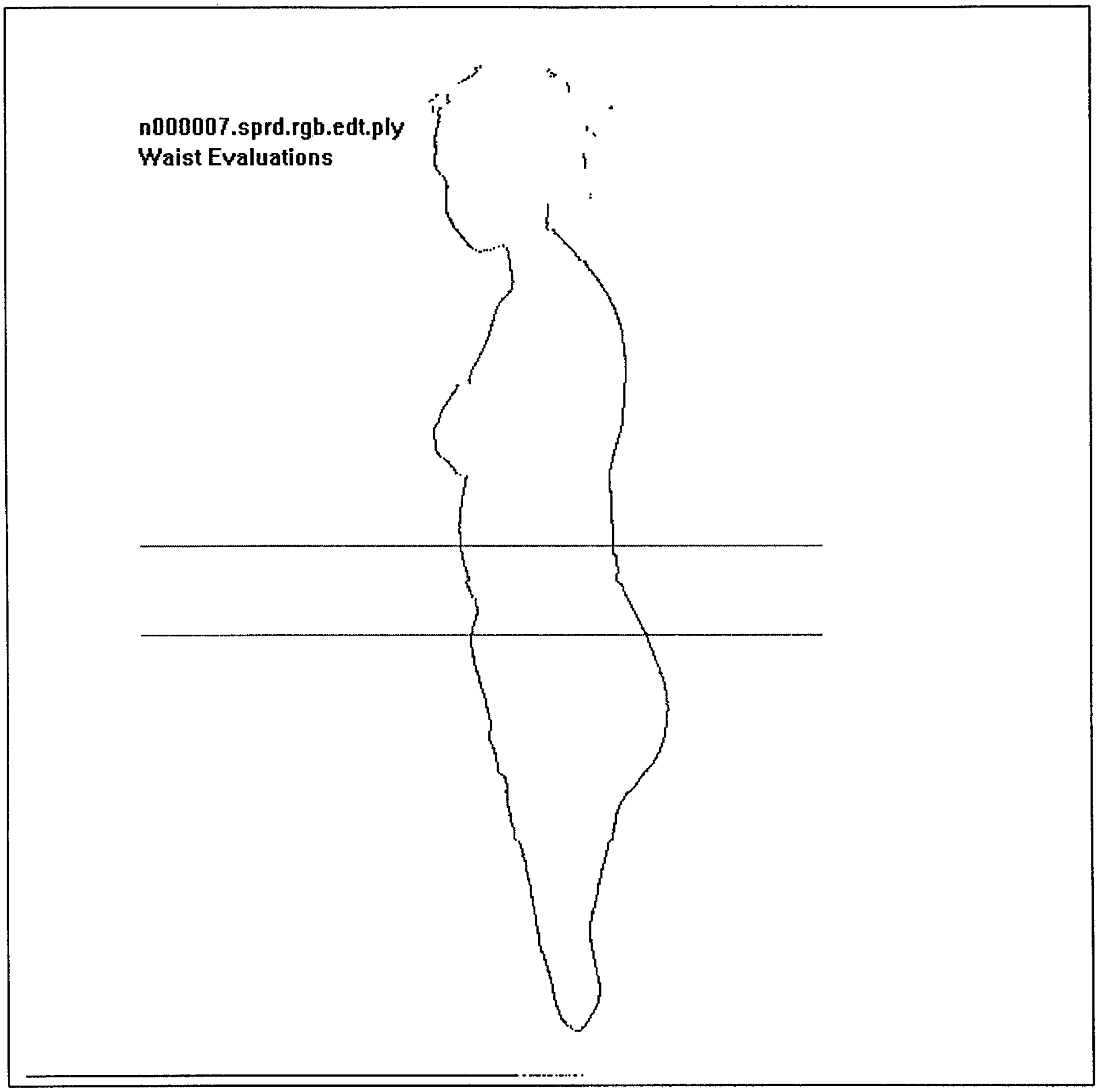

n000006.sprd.rgb.edt.ply
Crotch Height Range

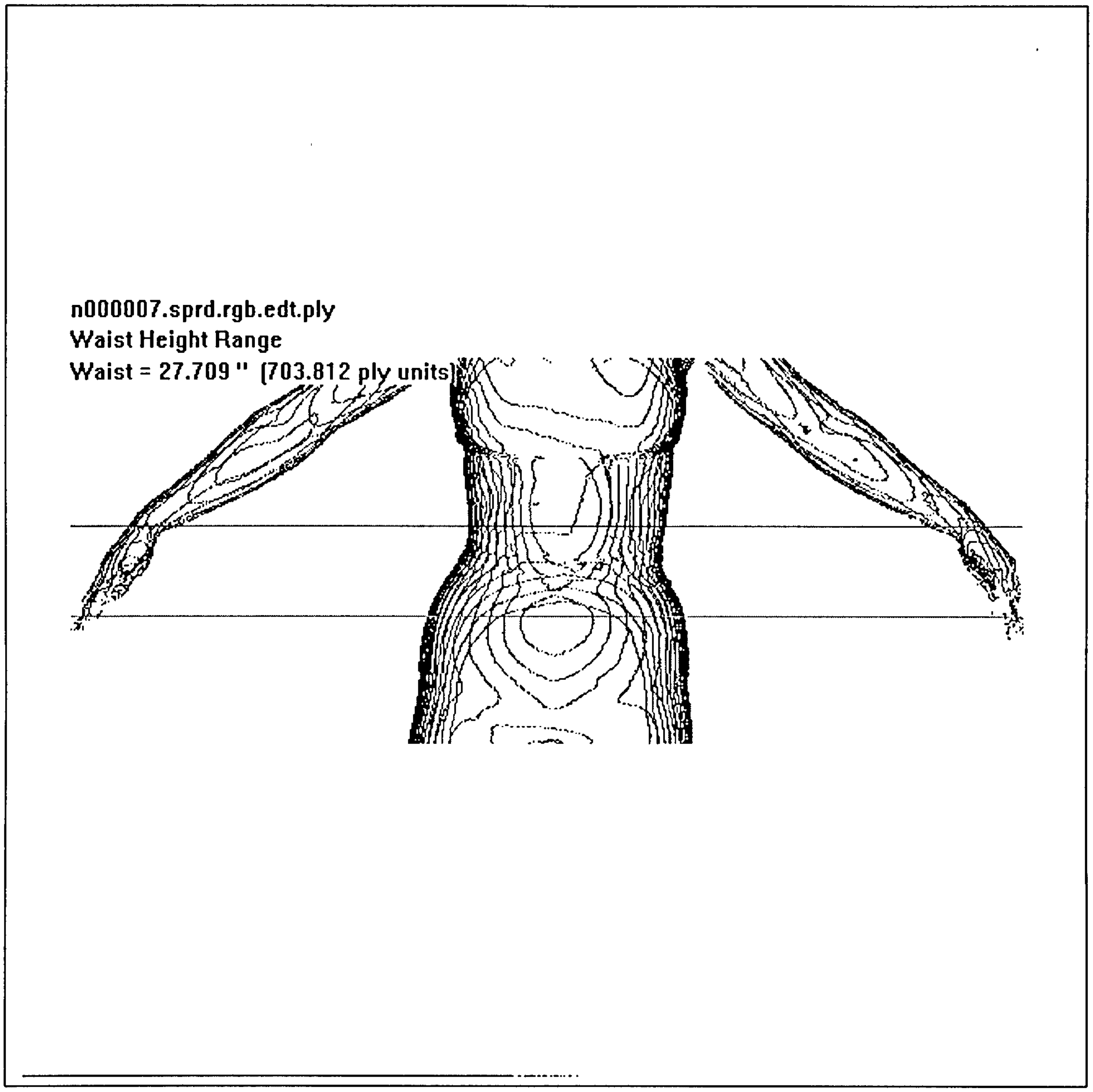

n000006.sprd.rgb.edt.ply
Waist To Floor

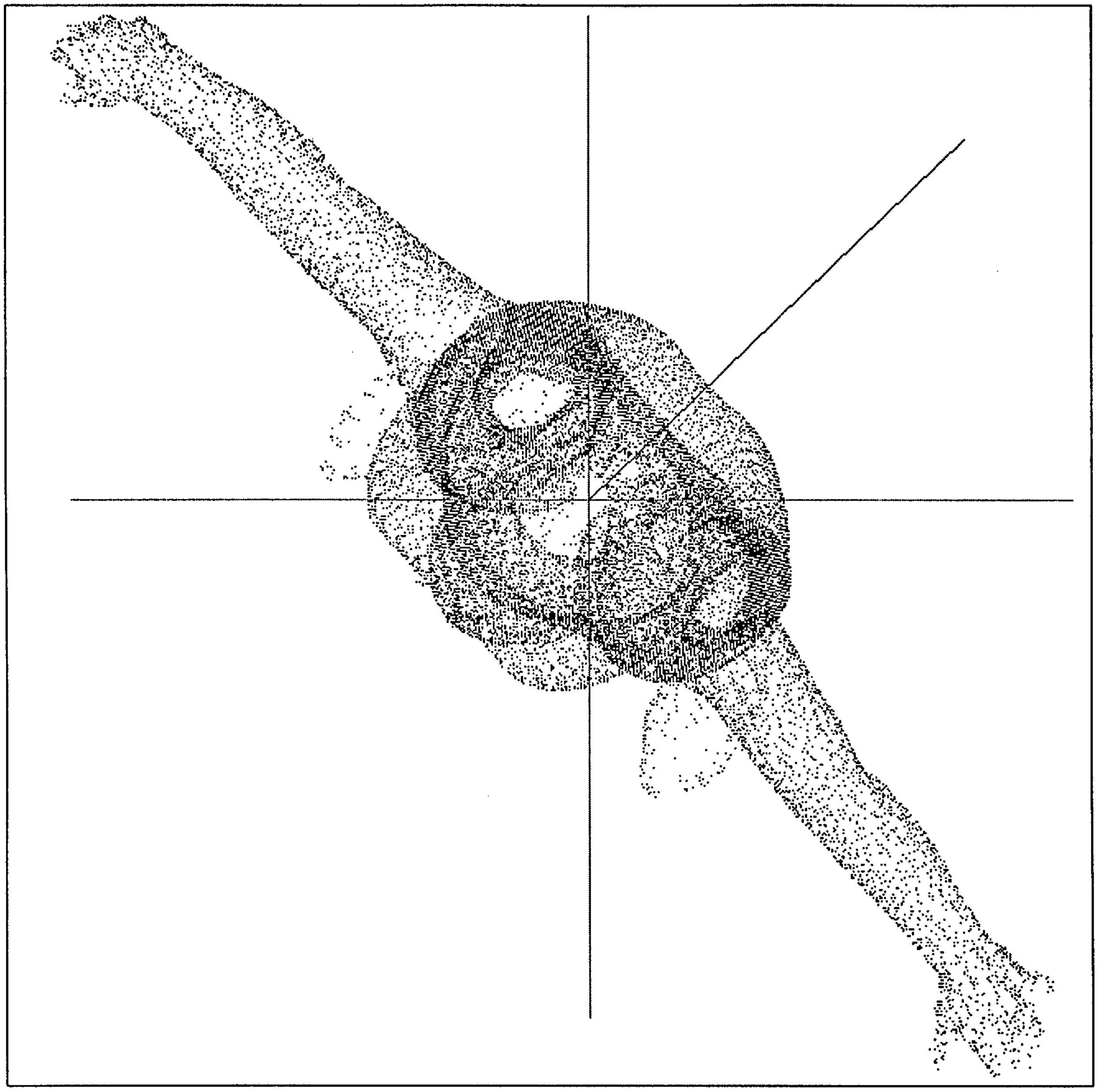

n000007.sprd.rgb.edt.ply
Chest Height Range
Chest = 36.337 " (922.966 ply units)

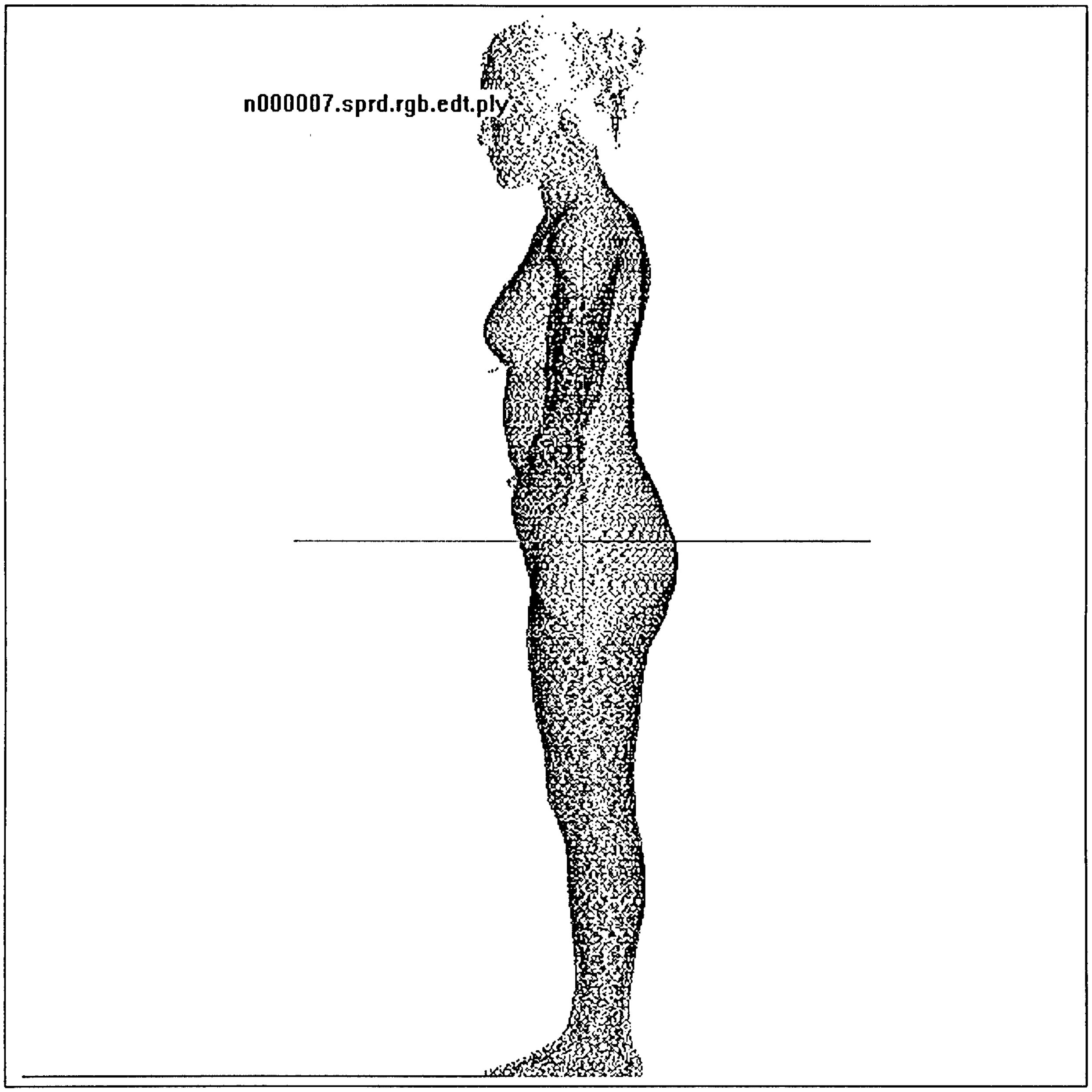

n000007.sprd.rgb.edt.ply
Waist To Floor

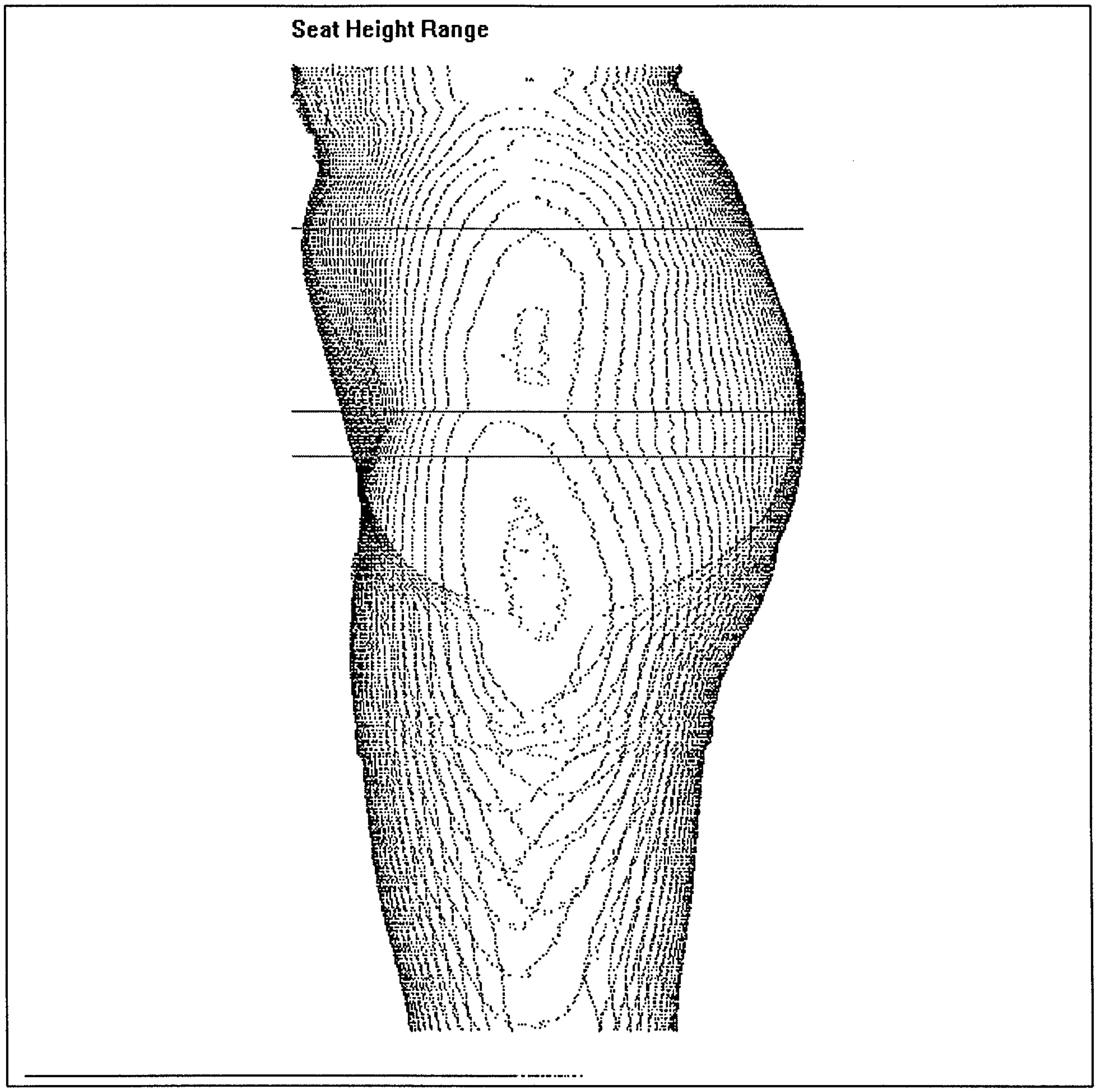

n000007.sprd.rgb.edt.ply
Waist To Floor

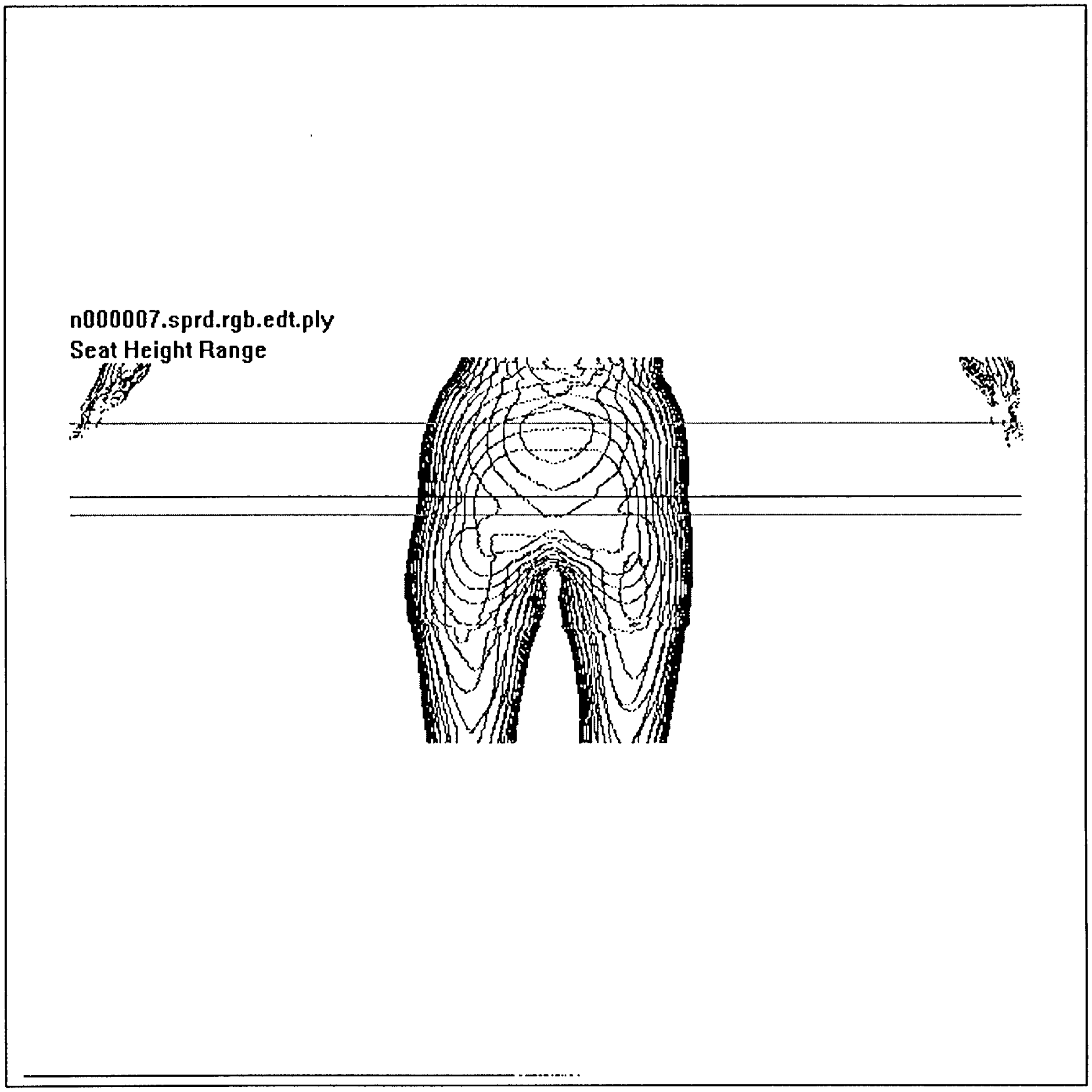

n000007.sprd.rgb.edt.ply
Waist Evaluations

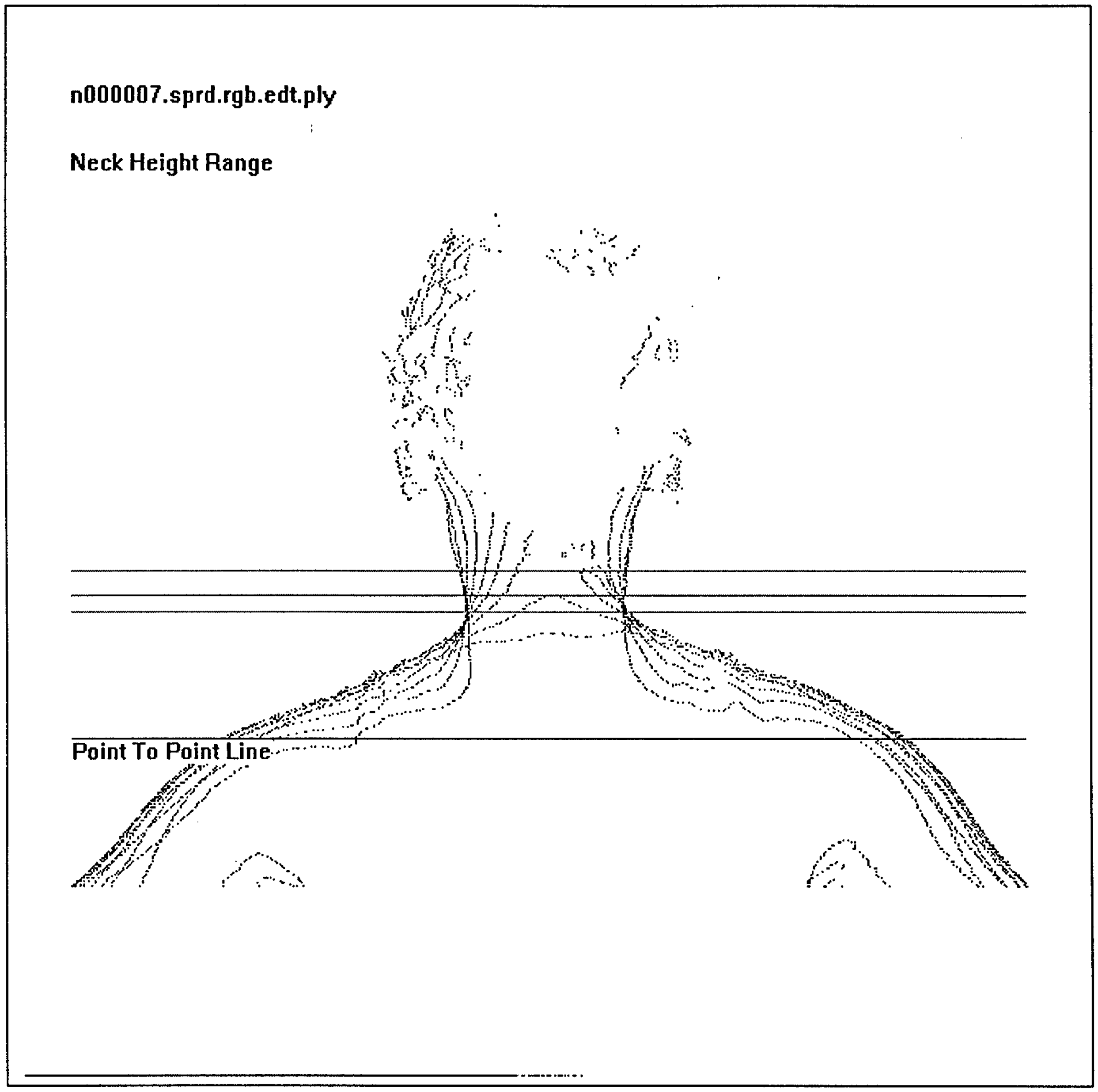

n000007.sprd.rgb.edt.ply
Waist Height Range
Waist = 27.709 " (703.812 ply units)

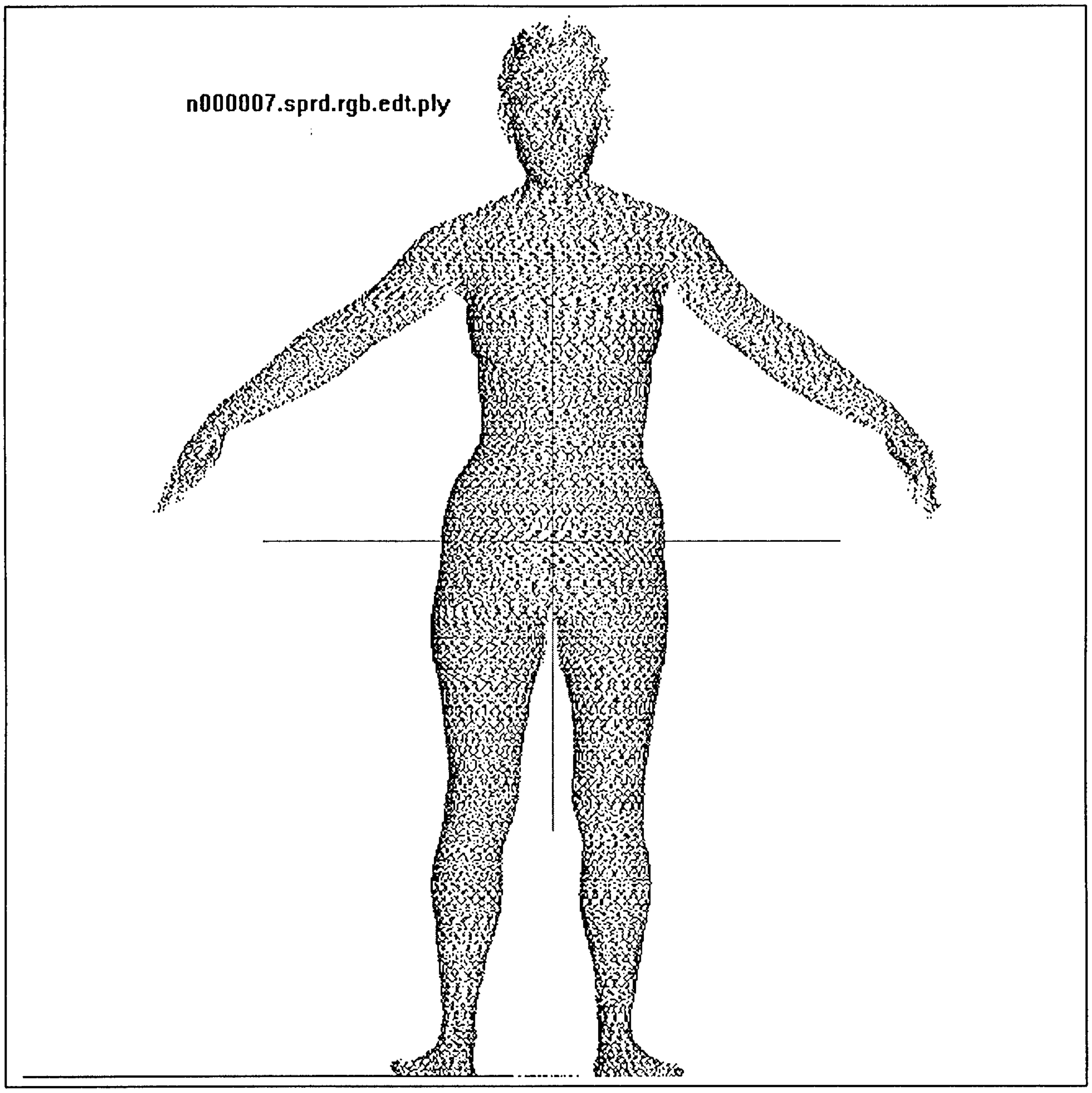


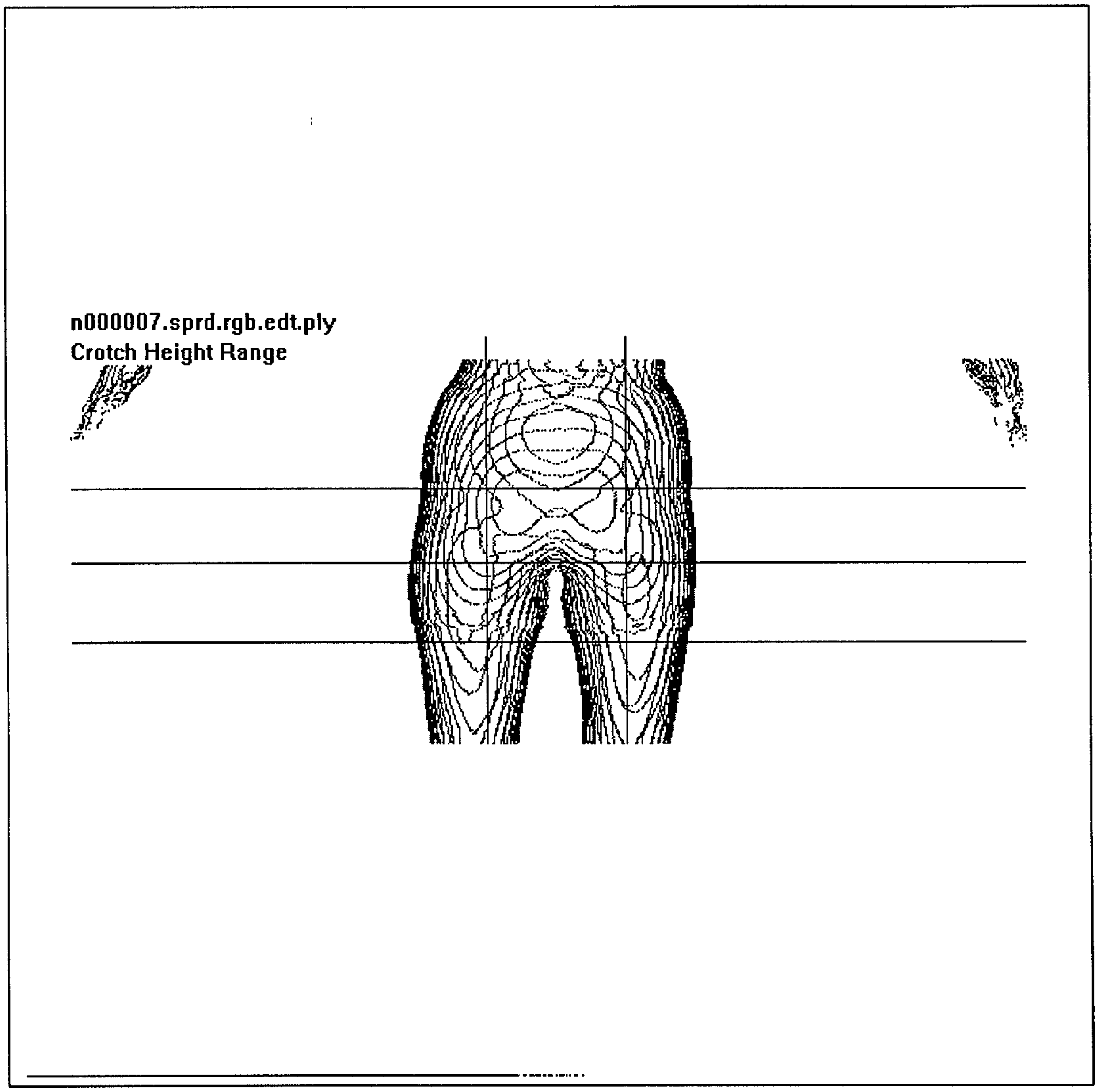

n000007.sprd.rgb.edt.ply

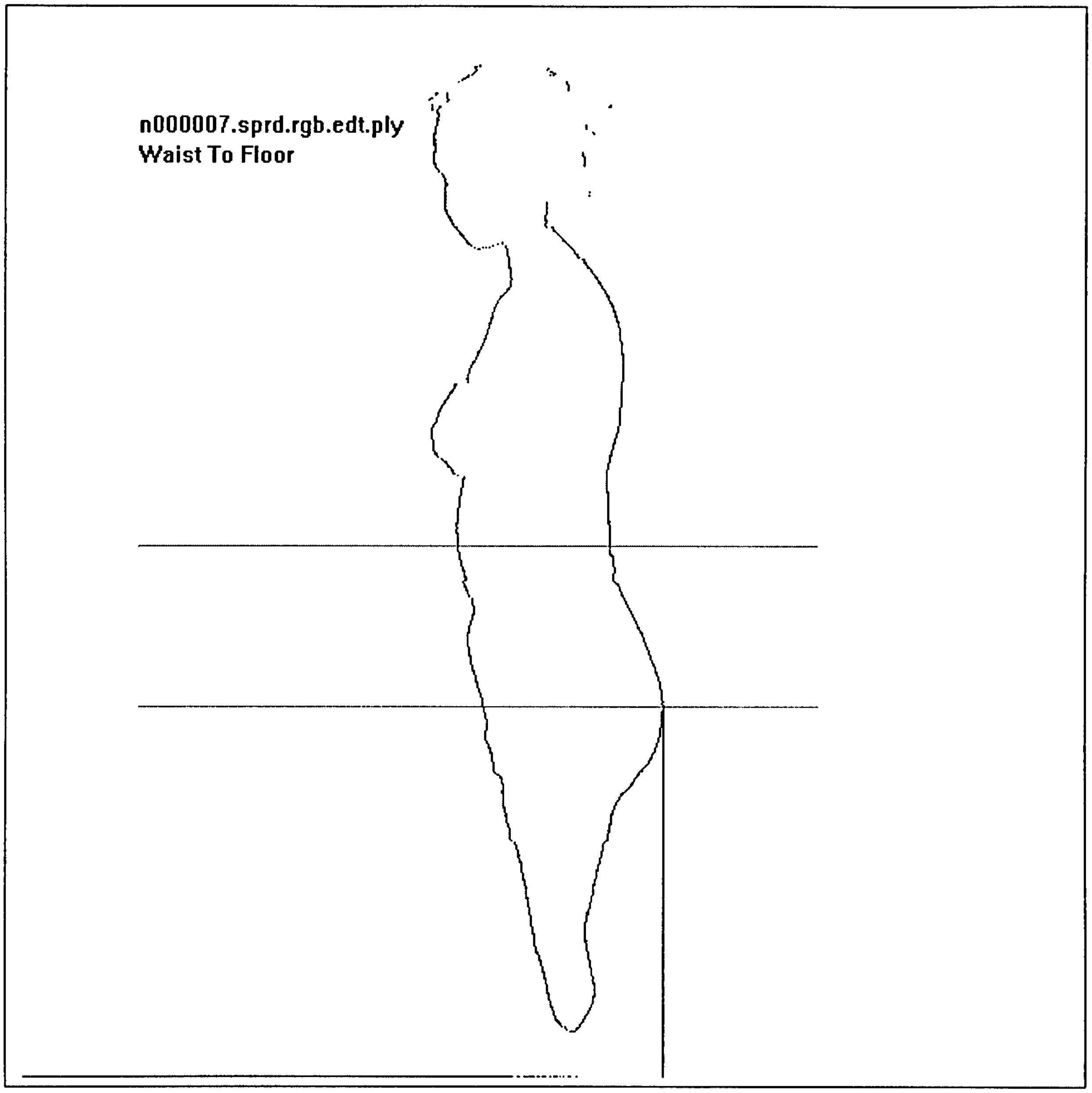

Seat Height Range

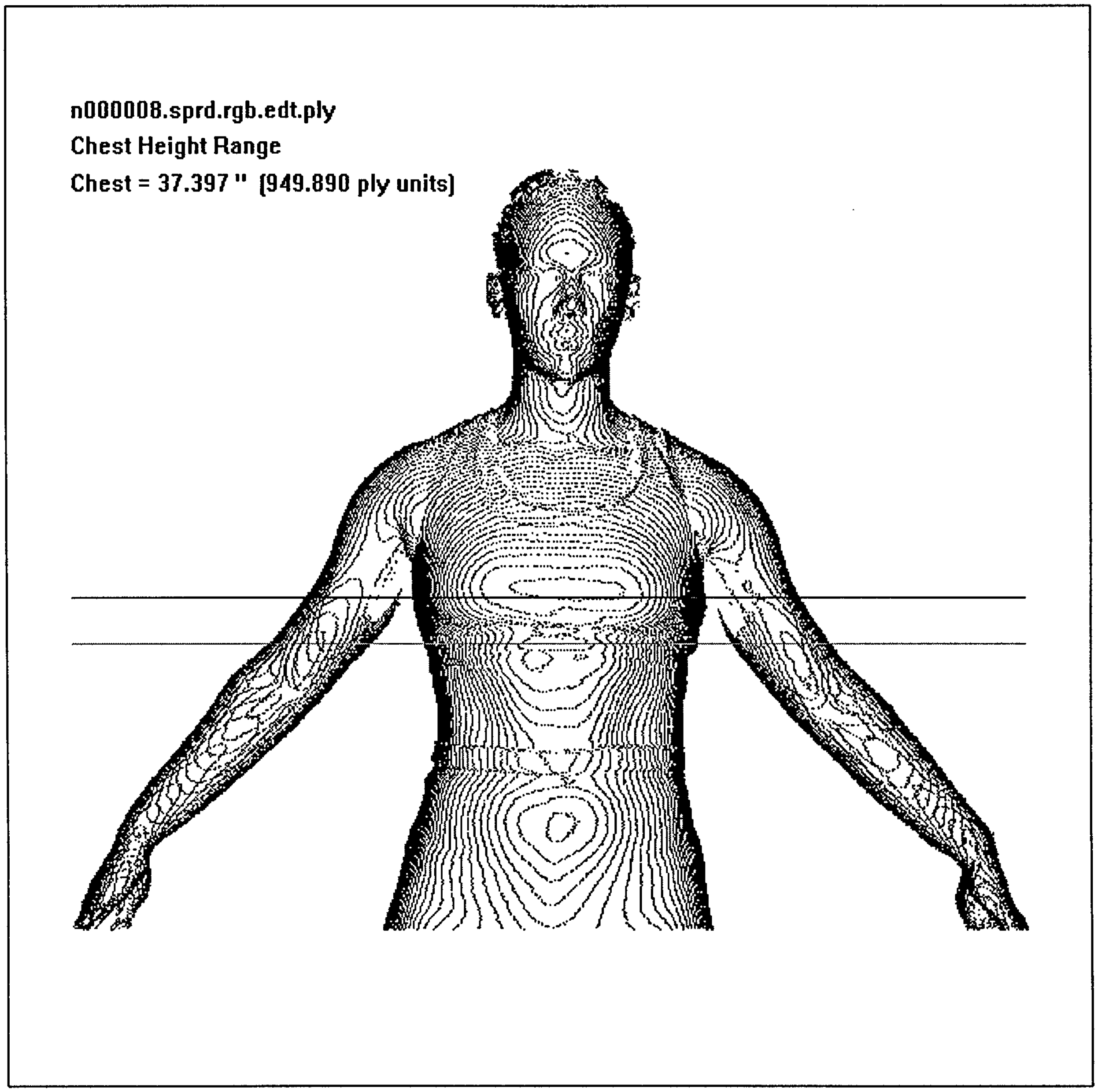

n000007.sprd.rgb.edt.ply
Seat Height Range


n000007.sprd.rgb.edt.ply
Neck Height Range
Point To Point Line


n000007.sprd.rgb.edt.ply

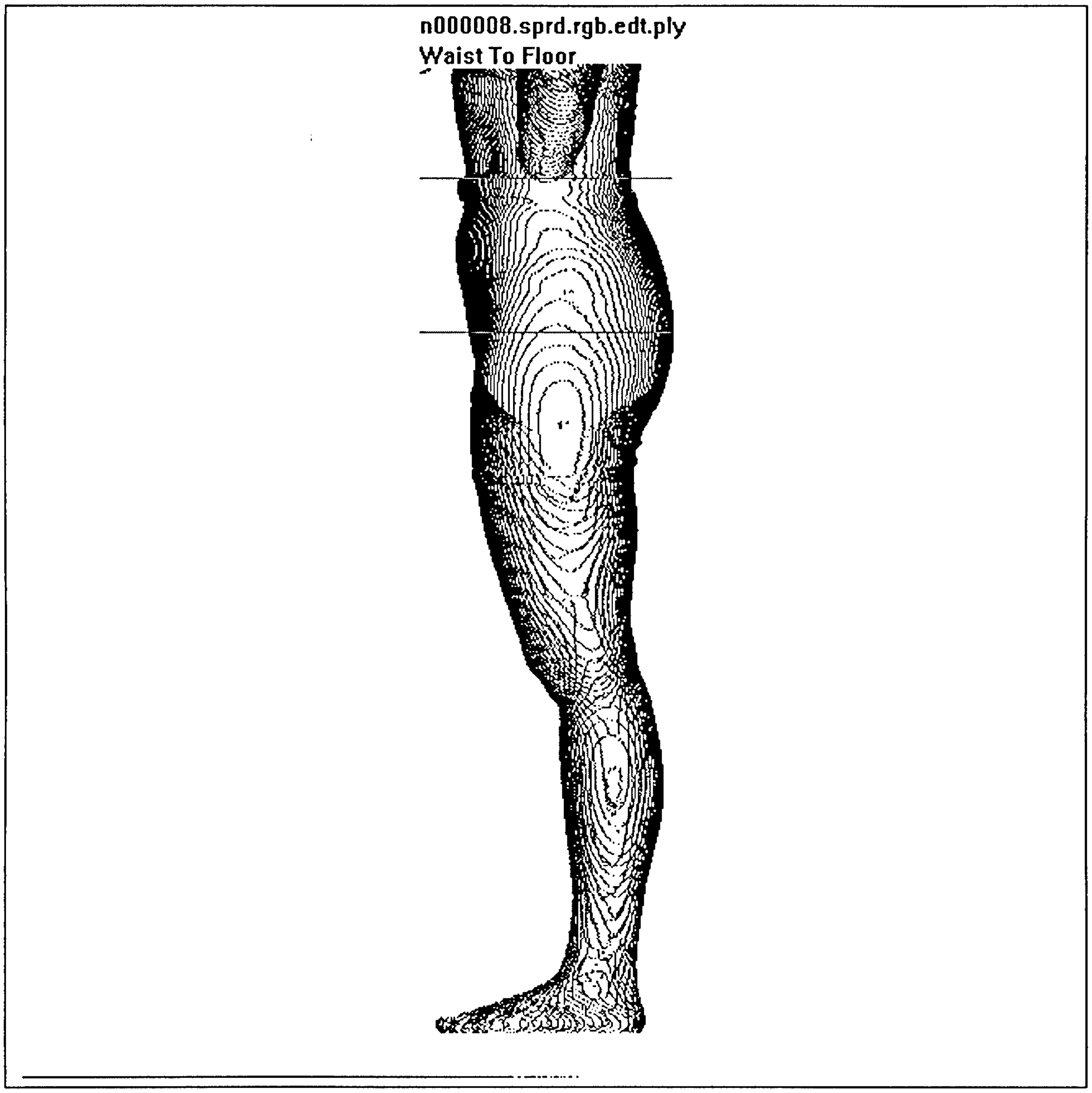

n000007.sprd.rgb.edt.ply
Crotch Height Range

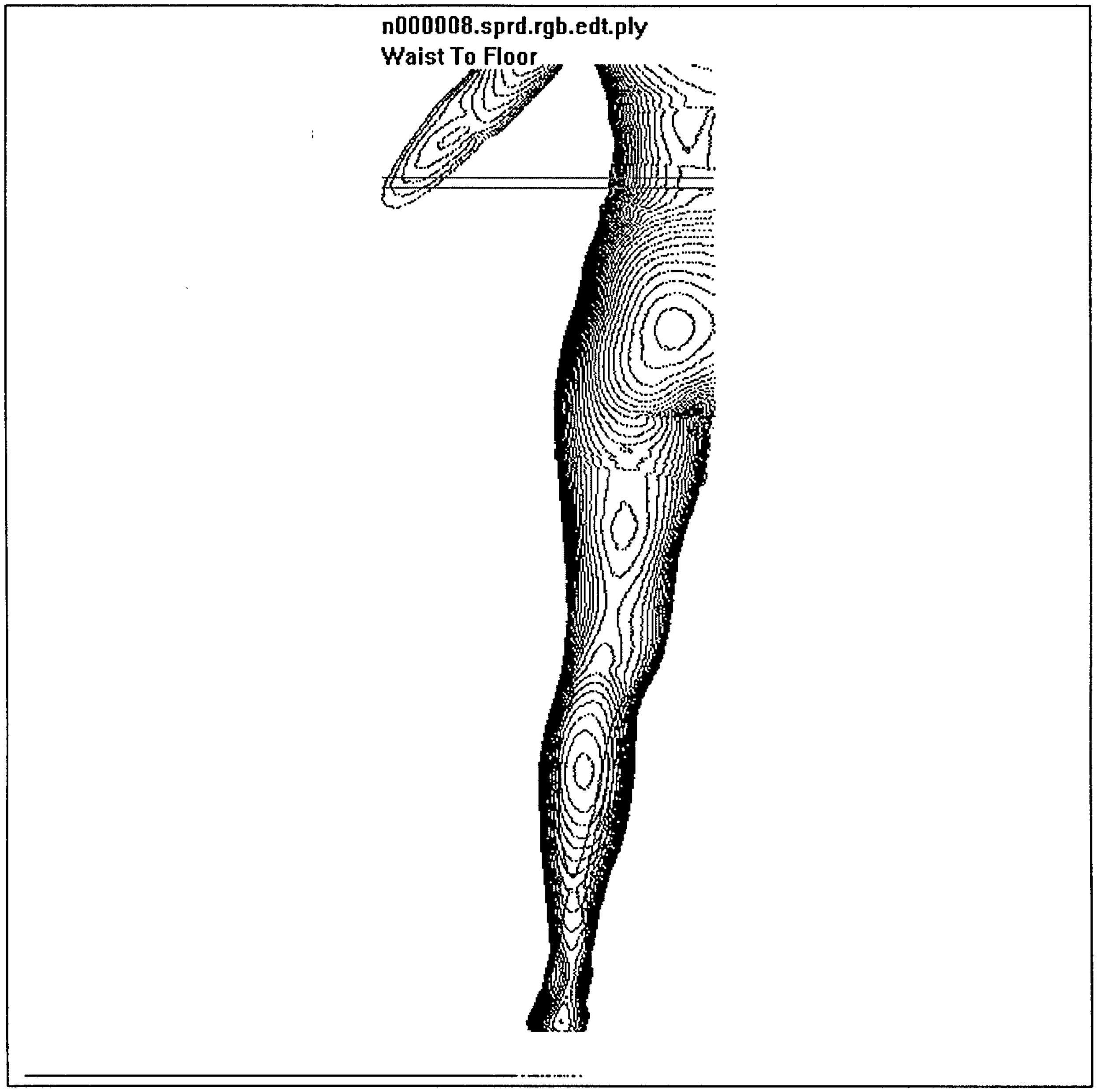

n000007.sprd.rgb.edt.ply
Waist To Floor

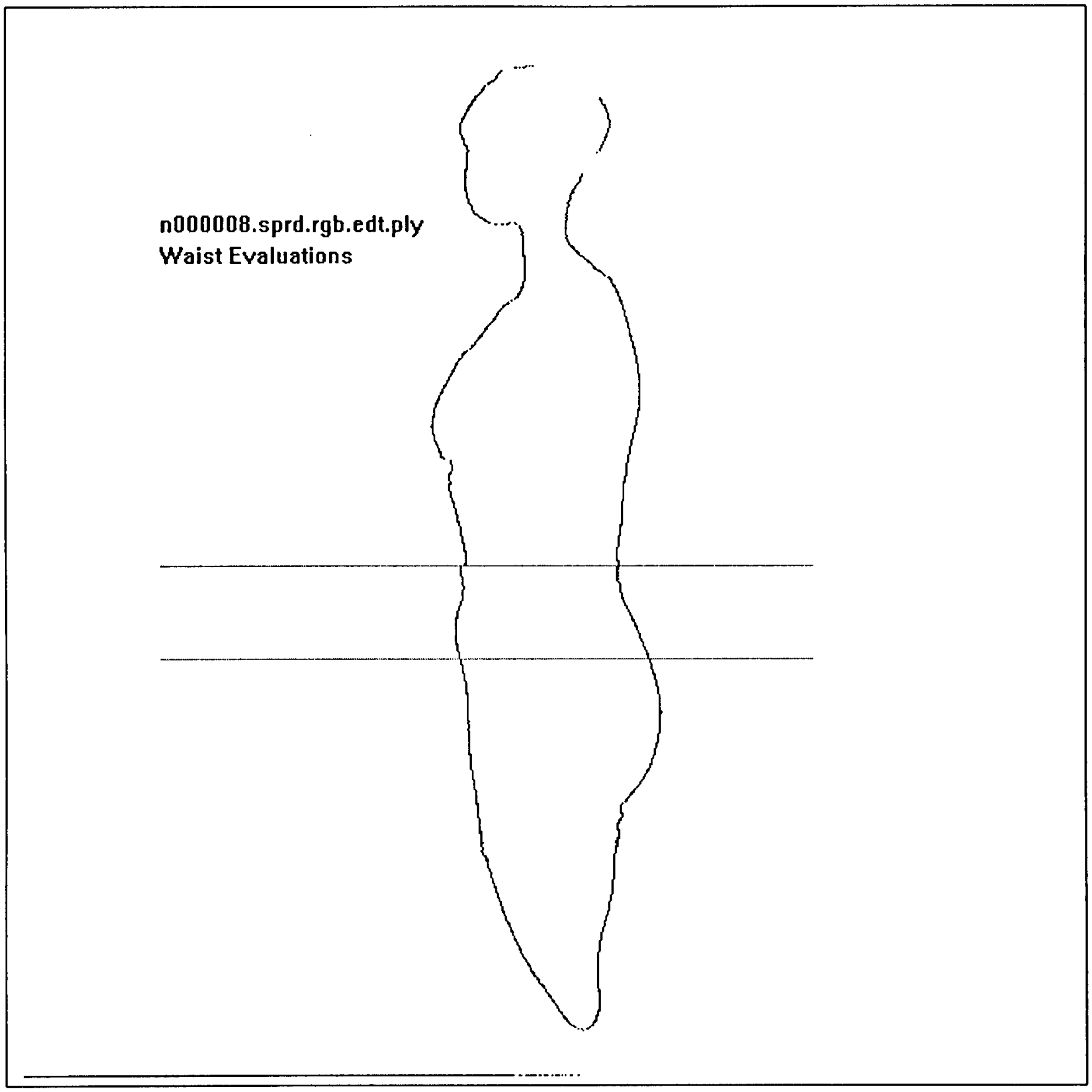

n000008.sprd.rgb.edt.ply

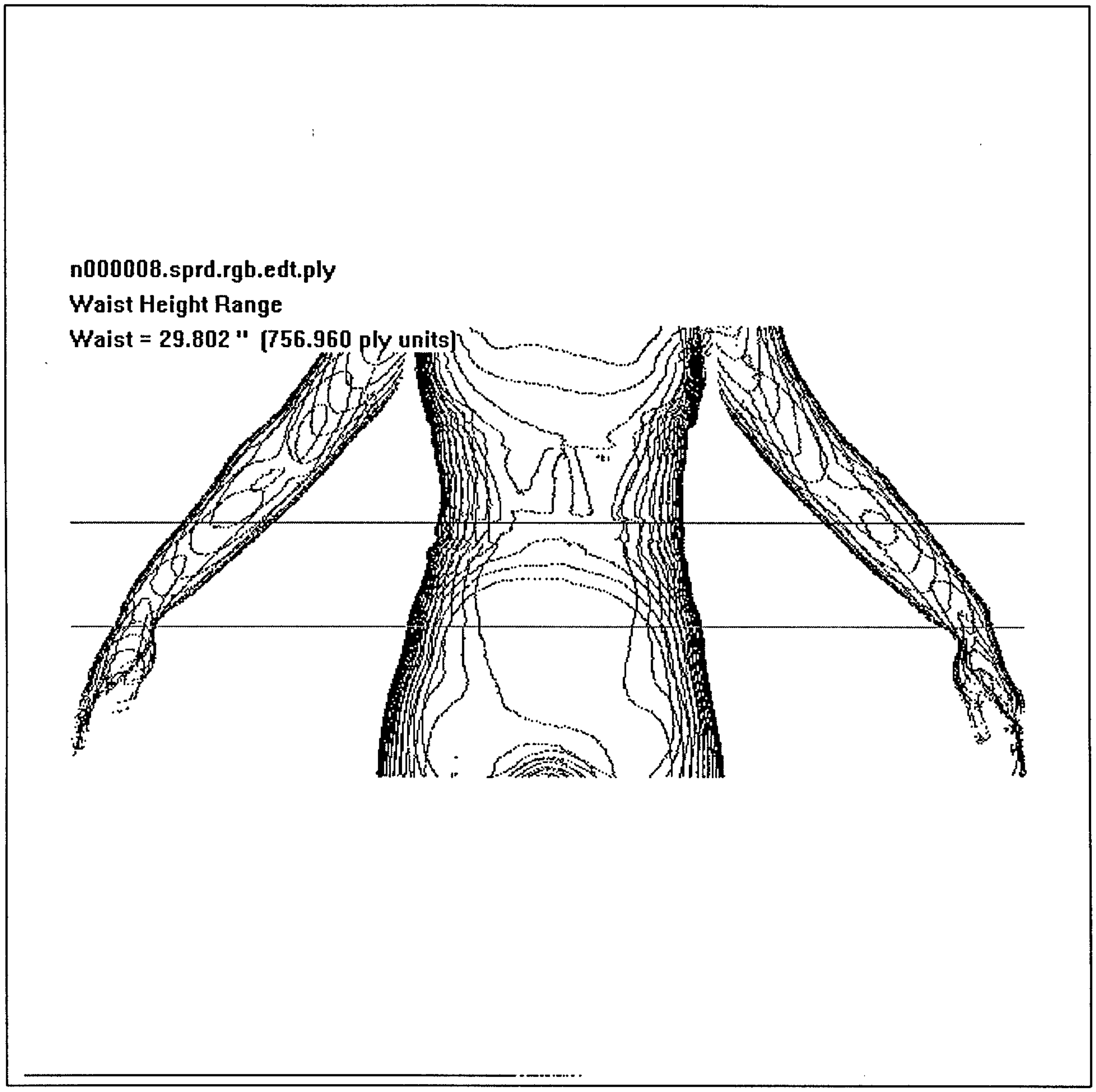
Chest Height Range

Chest = 37.397 " (949.890 ply units)


n000008.sprd.rgb.edt.ply
Waist To Floor


n000008.sprd.rgb.edt.ply
Waist To Floor

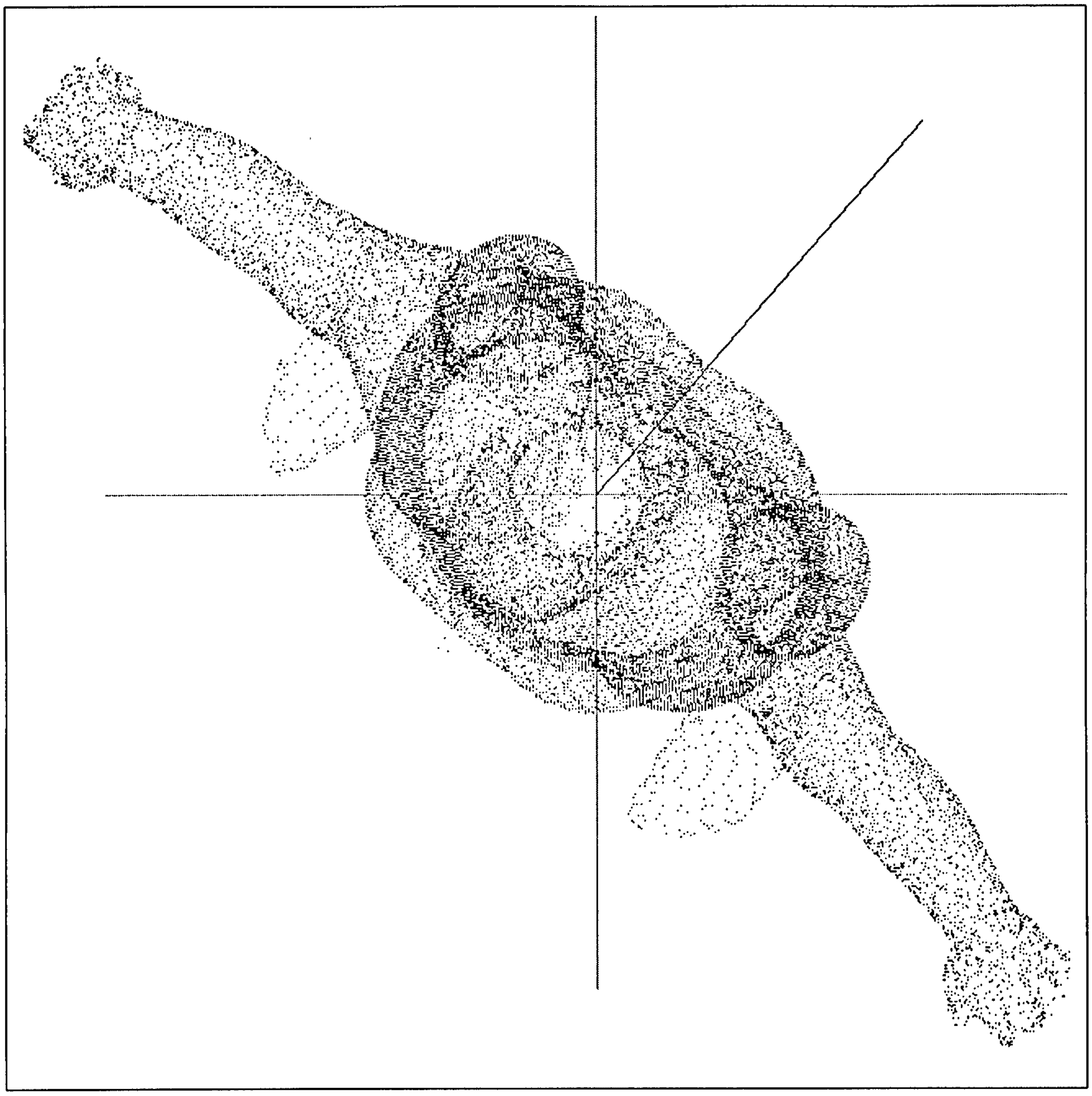

n000008.sprd.rgb.edt.ply
Waist Evaluations

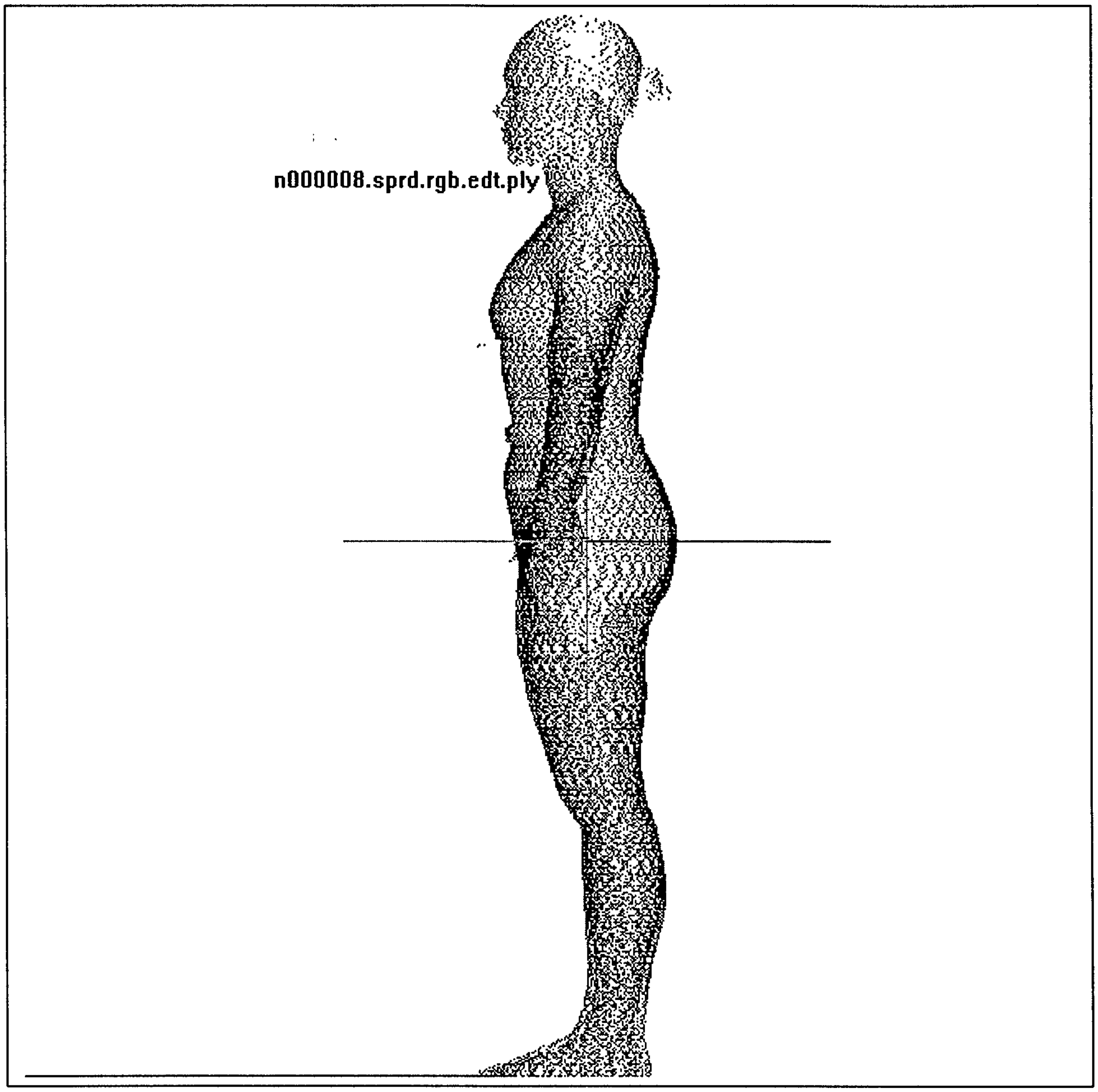


n000008.sprd.rgb.edt.ply

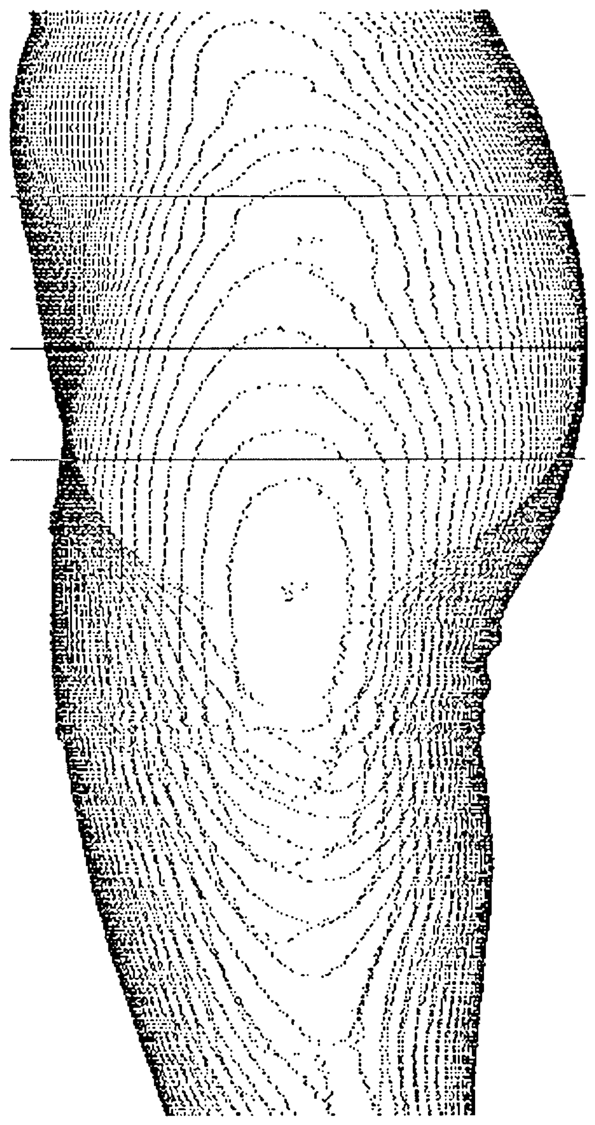
Waist Height Range

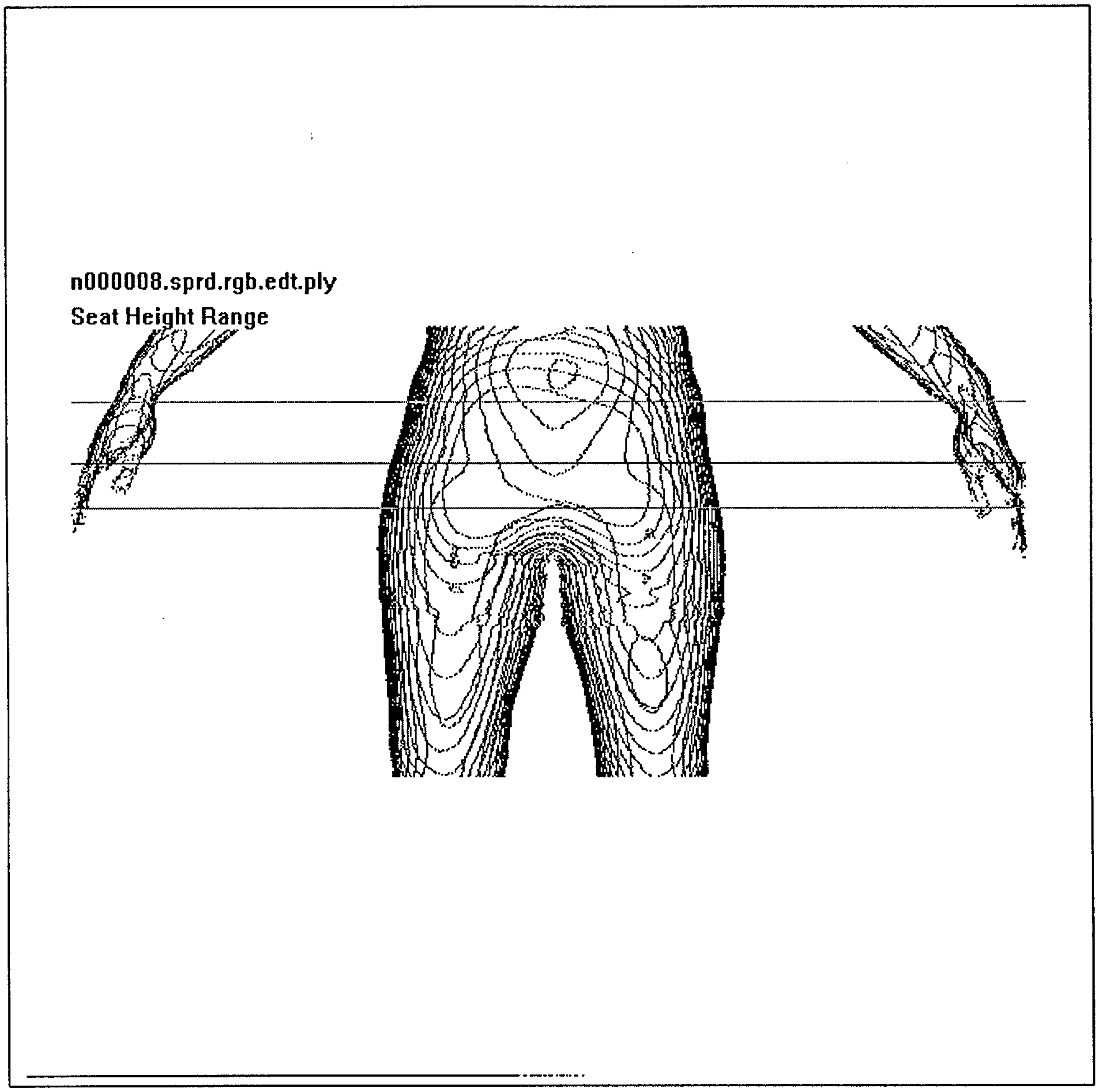
Waist = 29.802 " [756.960 ply units]

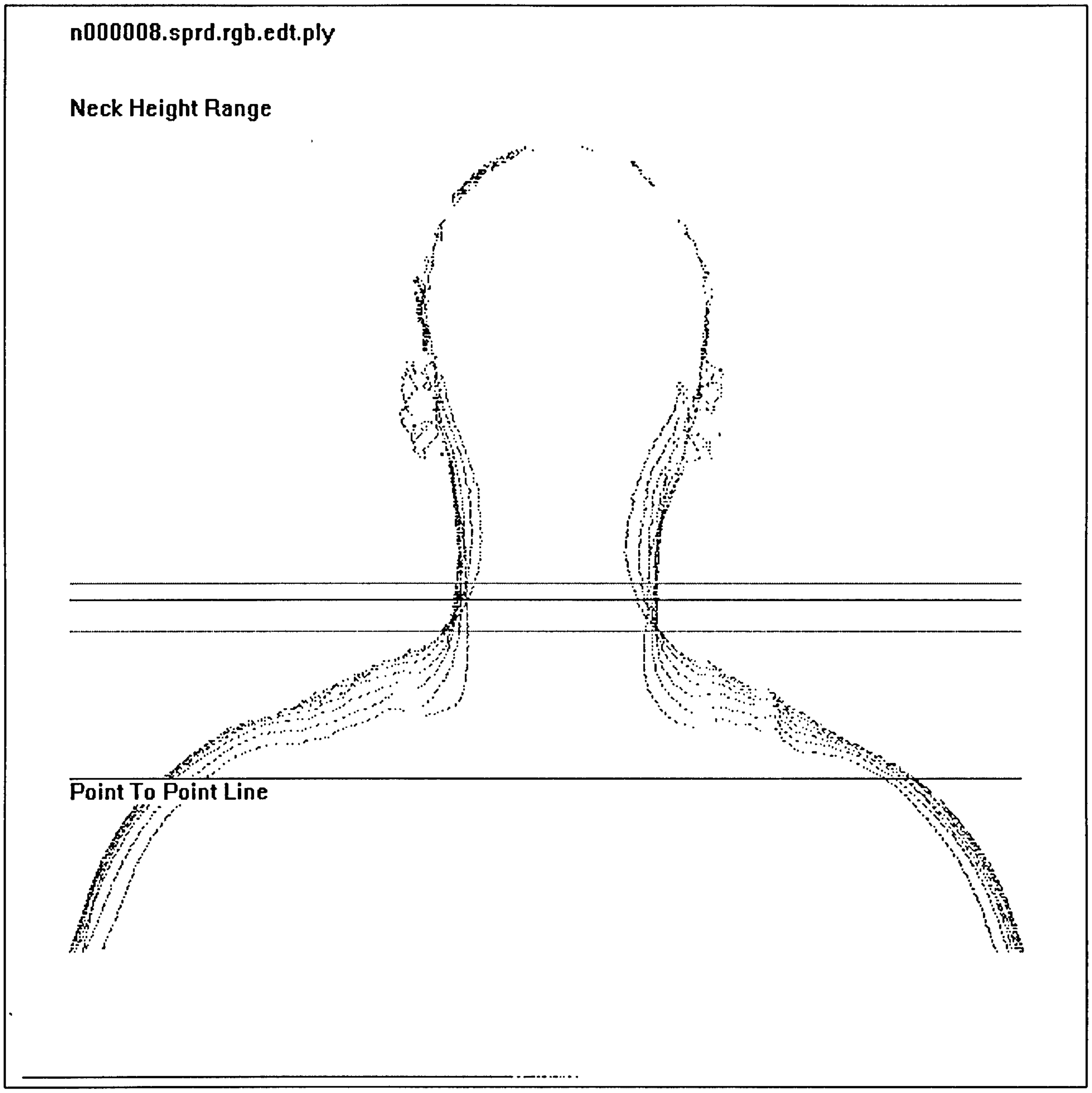


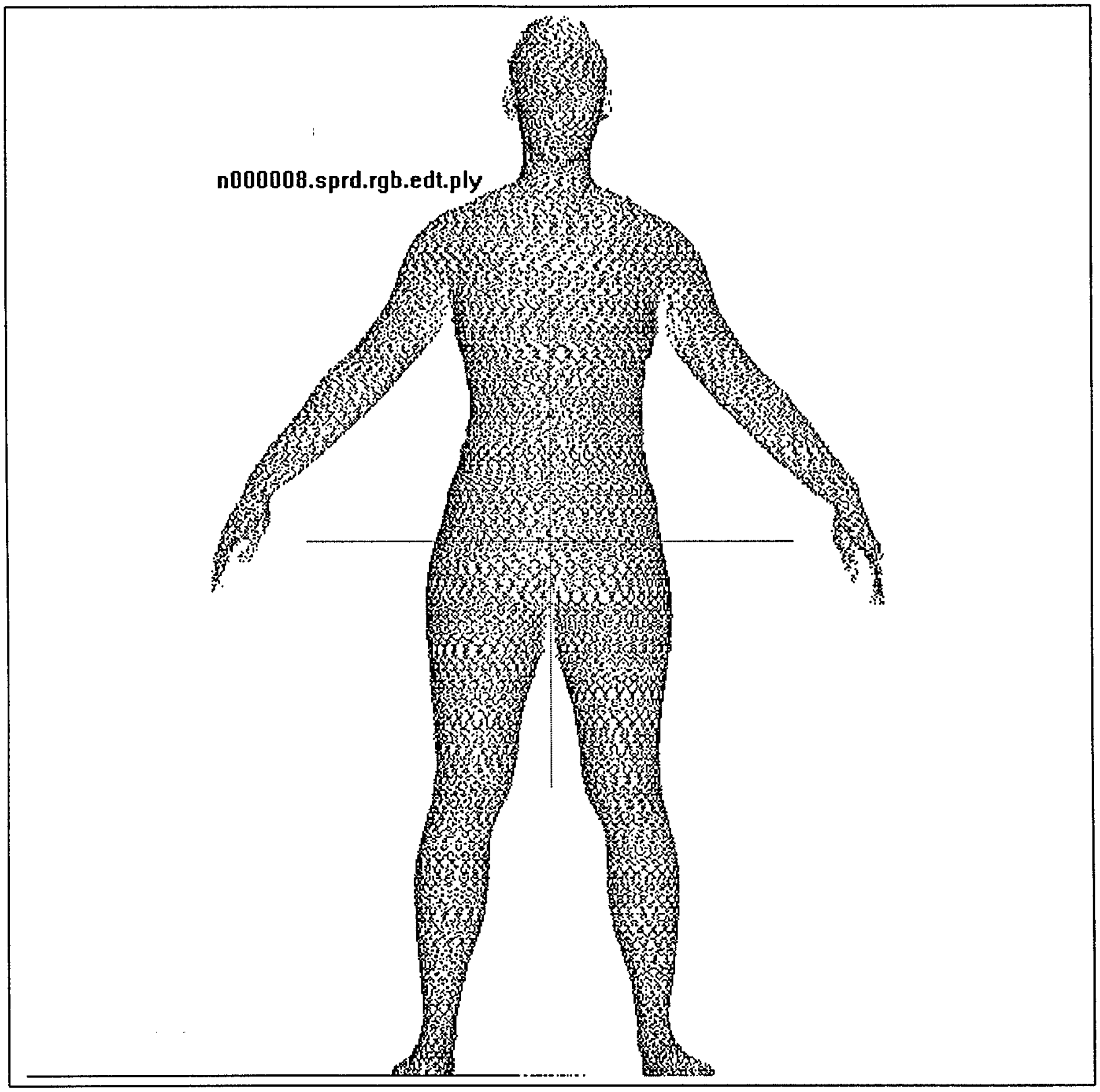

n000008.sprd.rgb.edt.ply

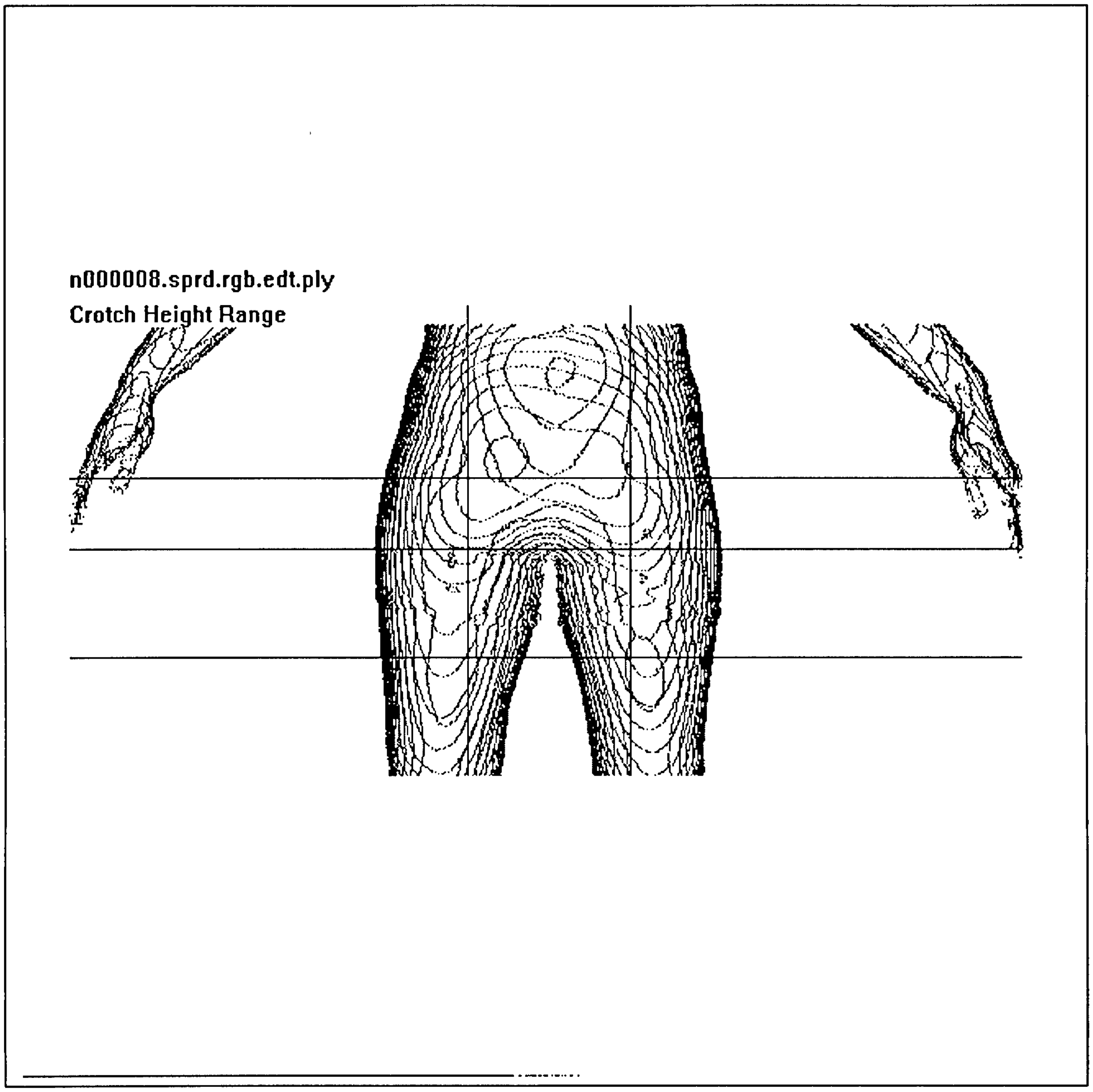
Seat Height Range

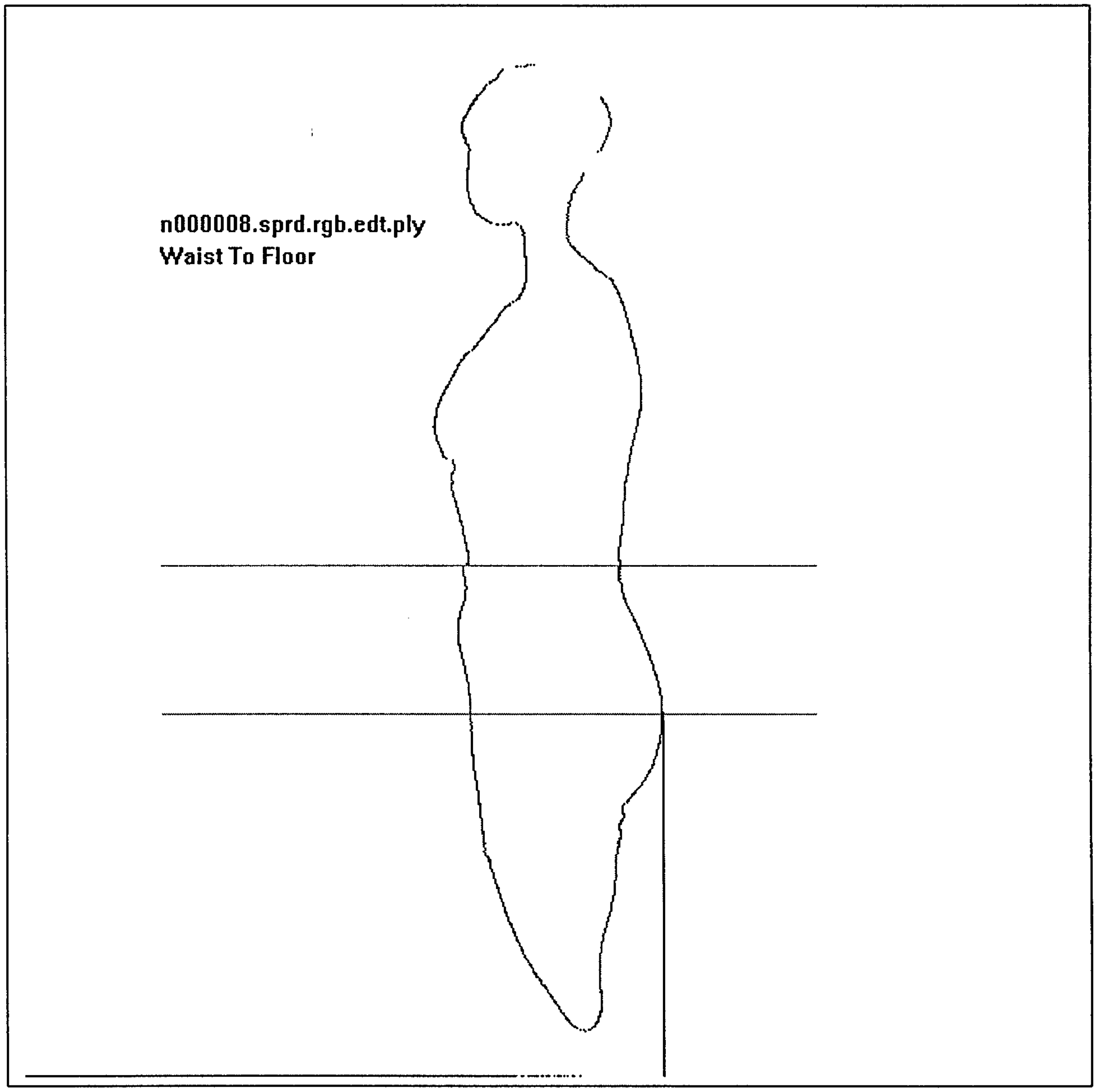


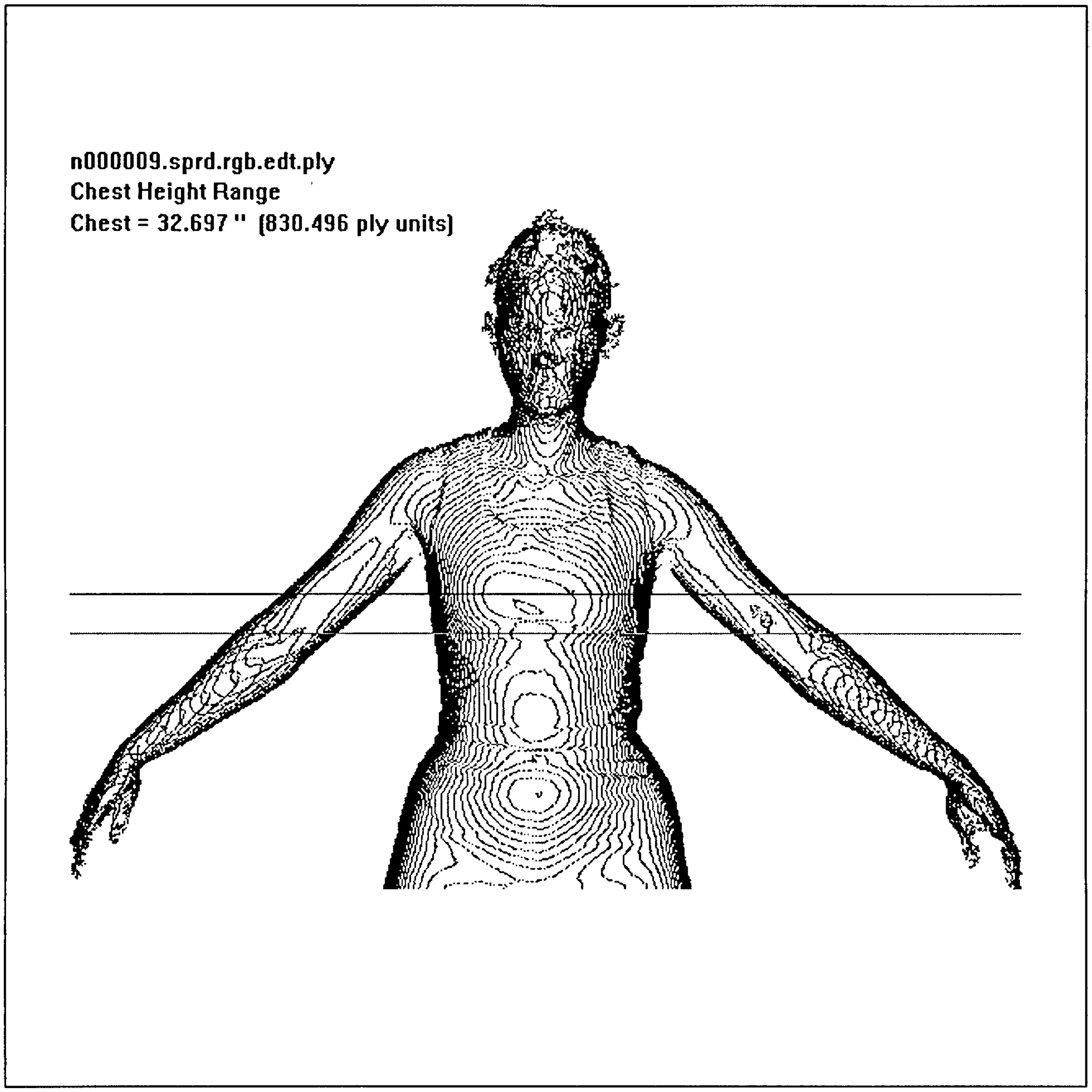

n000008.sprd.rgb.edt.ply
Seat Height Range


n000008.sprd.rgb.edt.ply
Neck Height Range
Point To Point Line

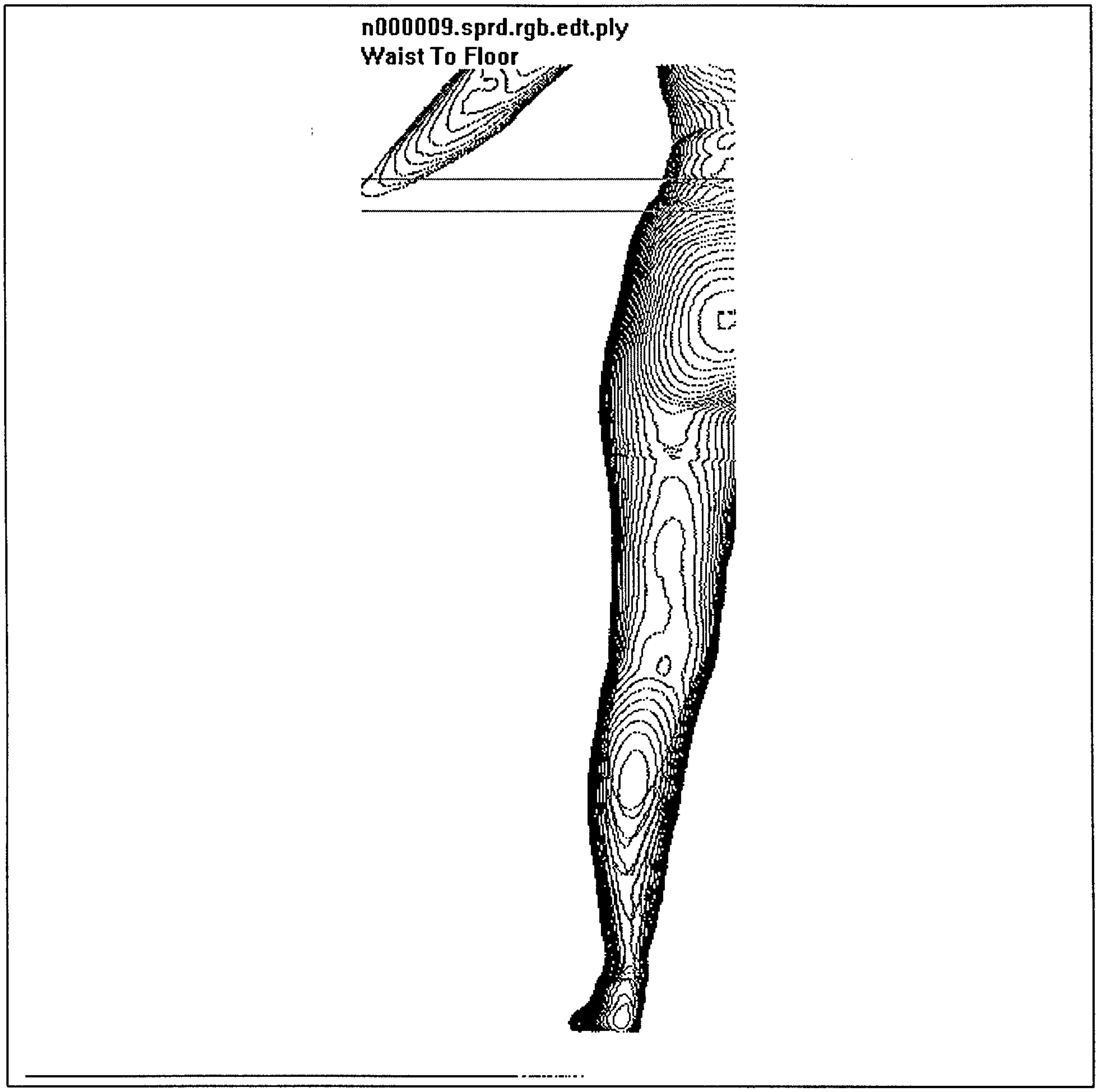

n000008.sprd.rgb.edt.ply

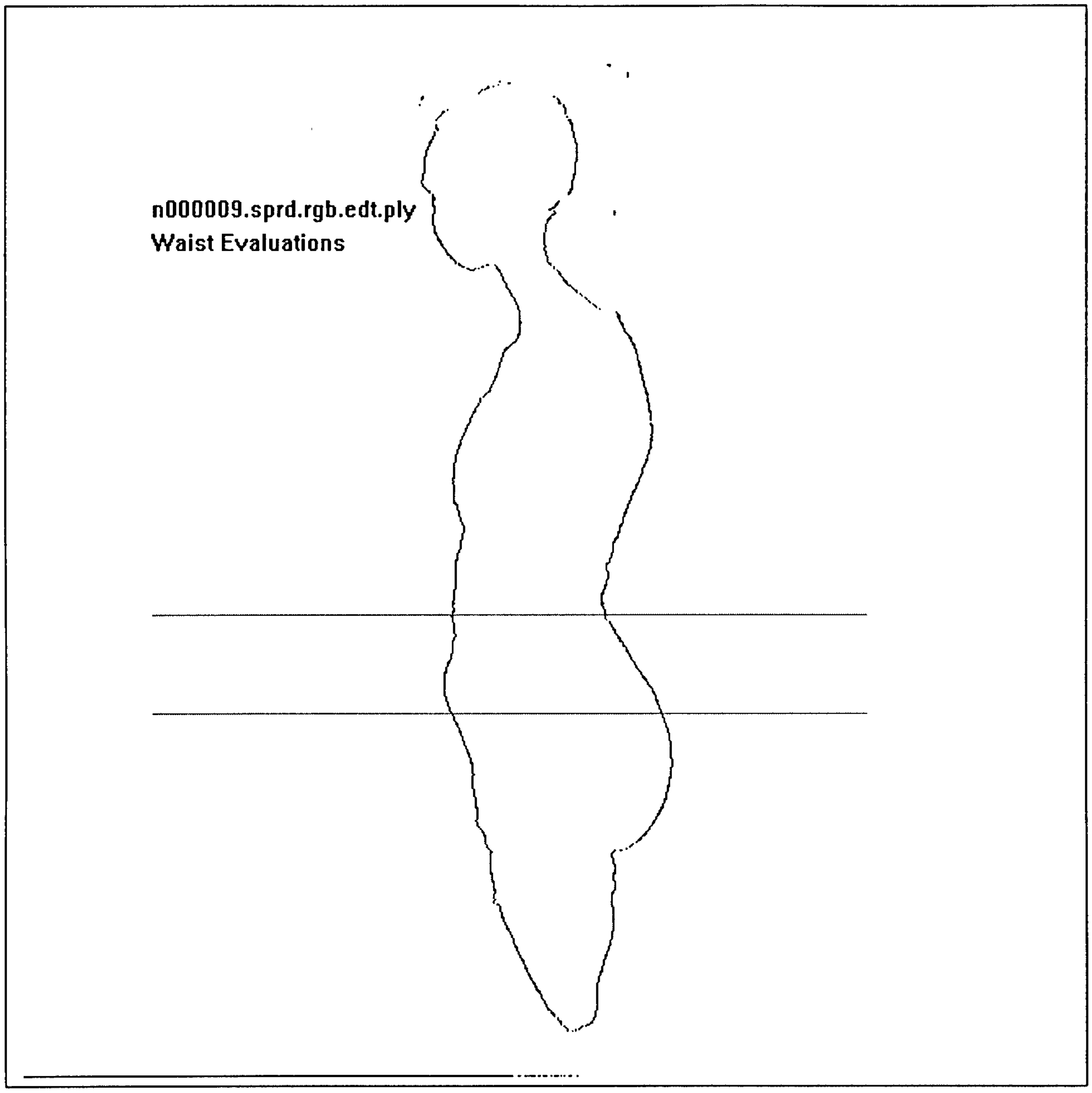

n000008.sprd.rgb.edt.ply
Crotch Height Range

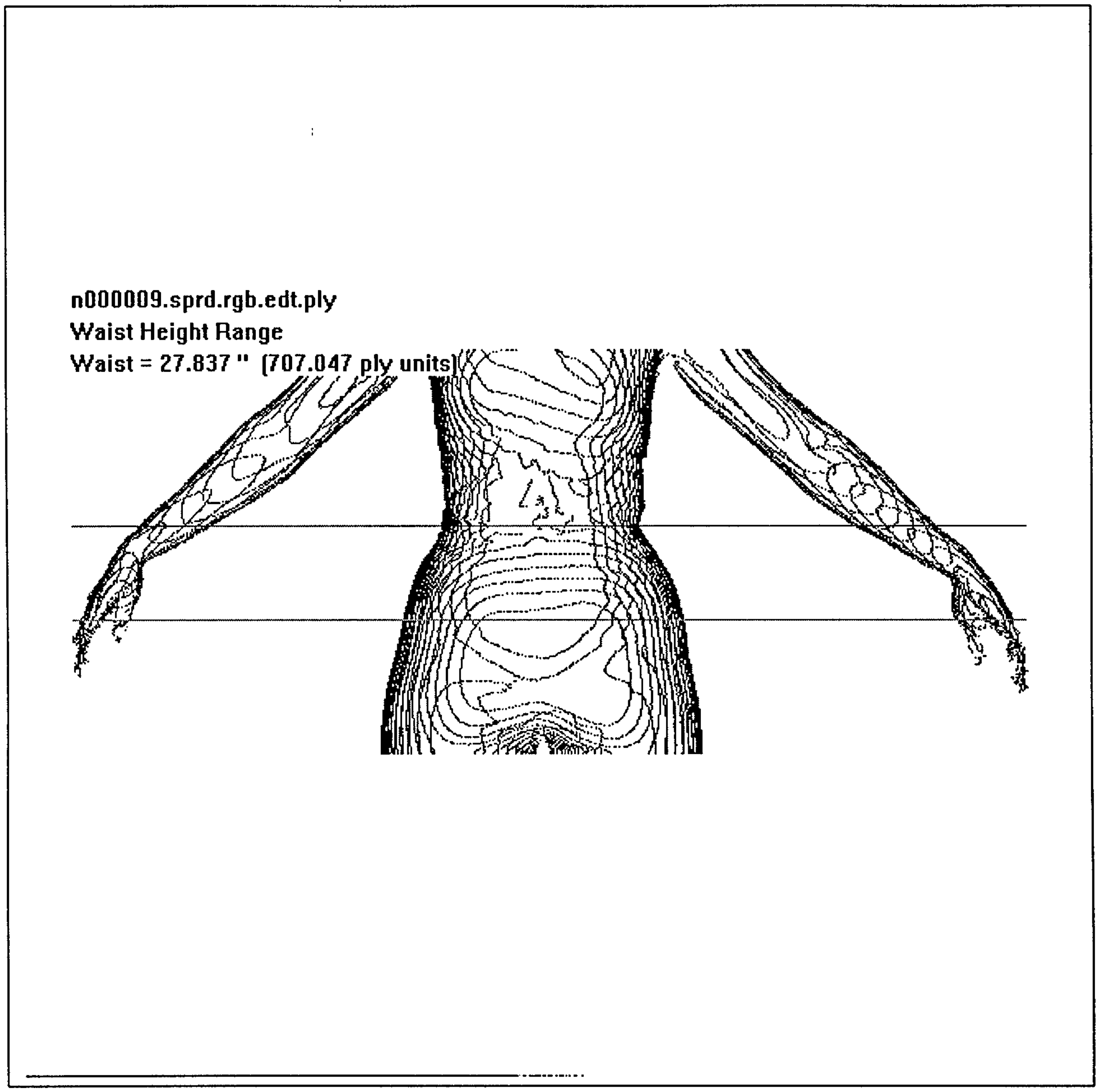

n000008.sprd.rgb.edt.ply
Waist To Floor



n000009.sprd.rgb.edt.ply
Chest Height Range
Chest = 32.697 " (830.496 ply units)


n000009.sprd.rgb.edt.ply
Waist To Floor

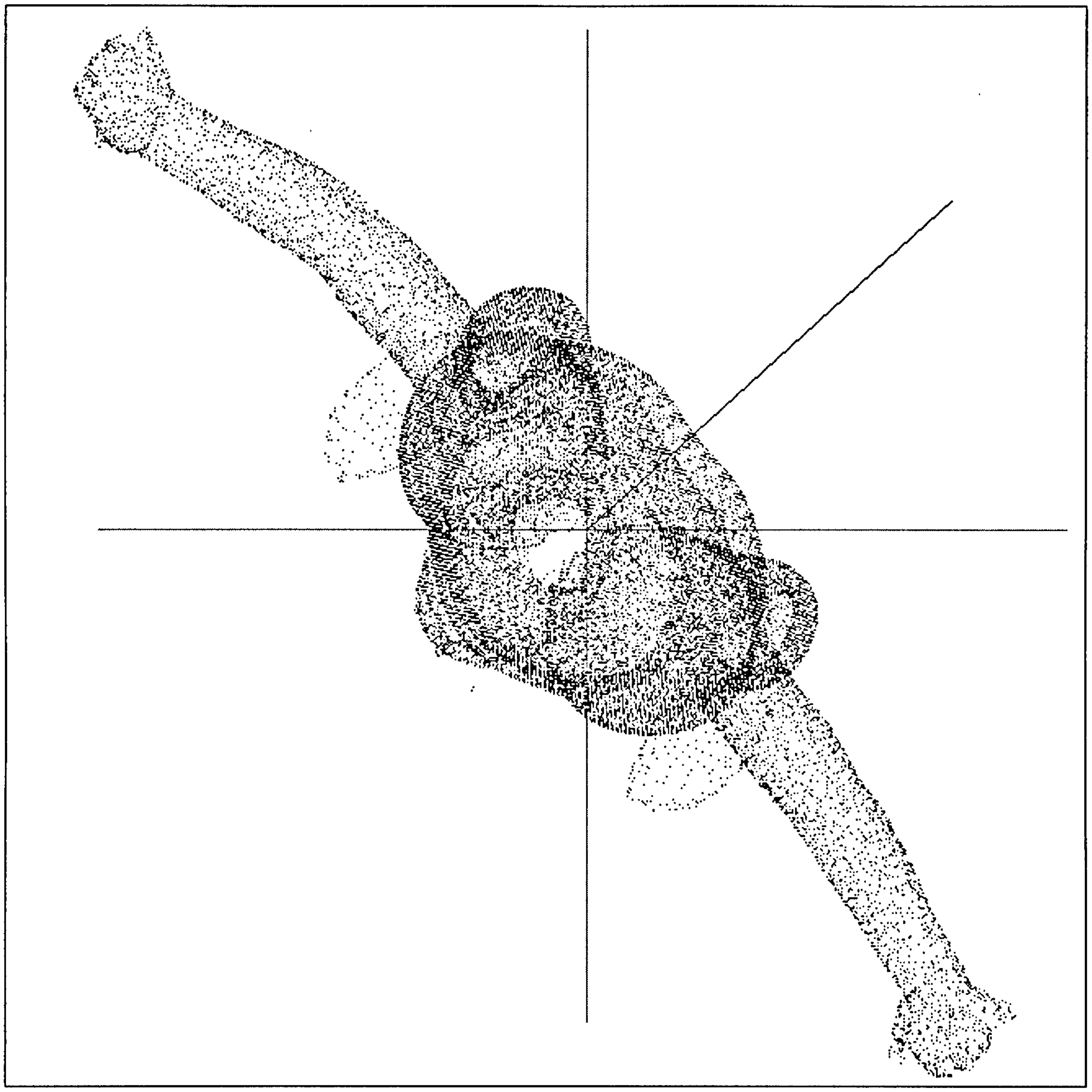

n000009.sprd.rgb.edt.ply
Waist Evaluations

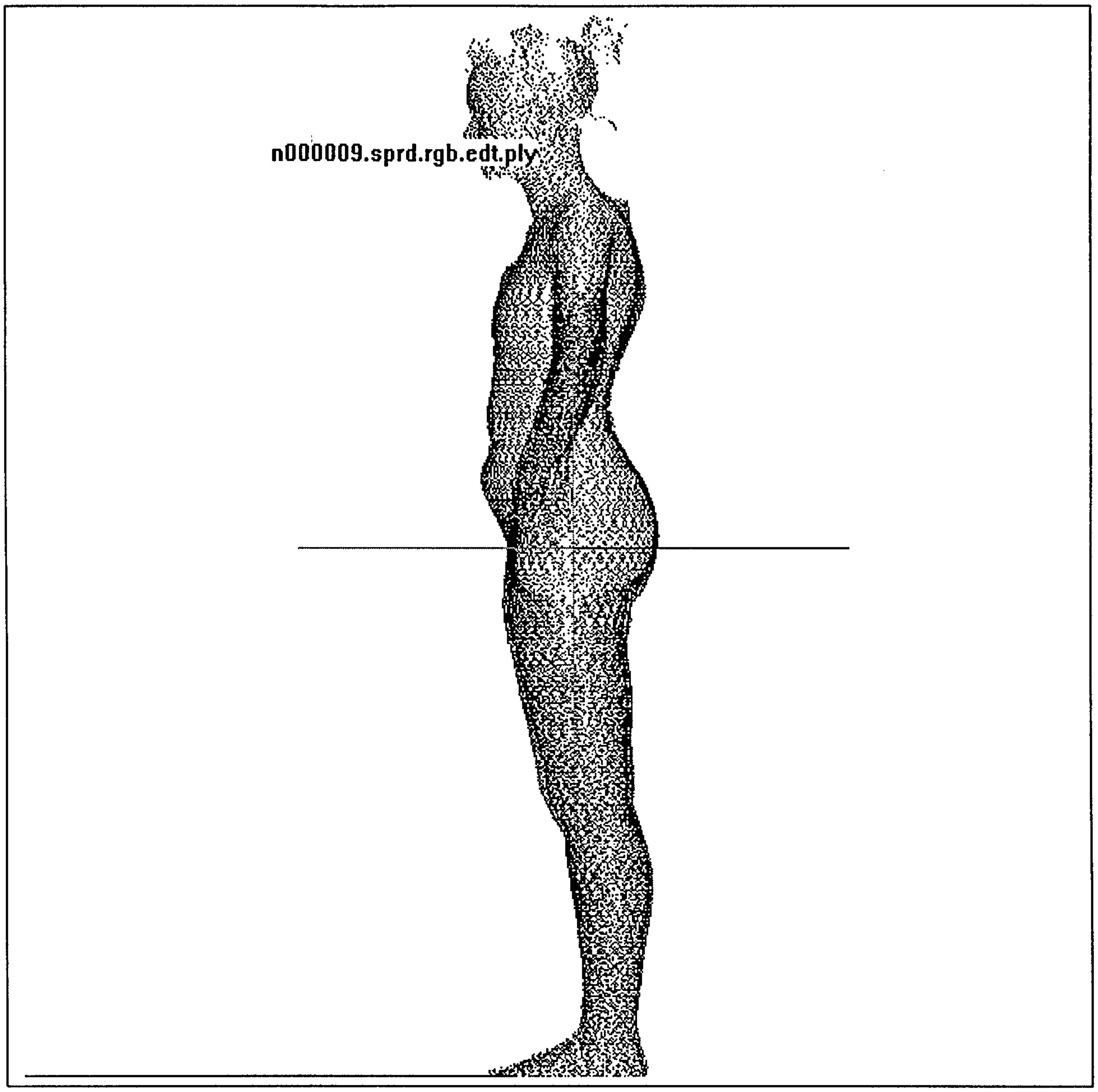


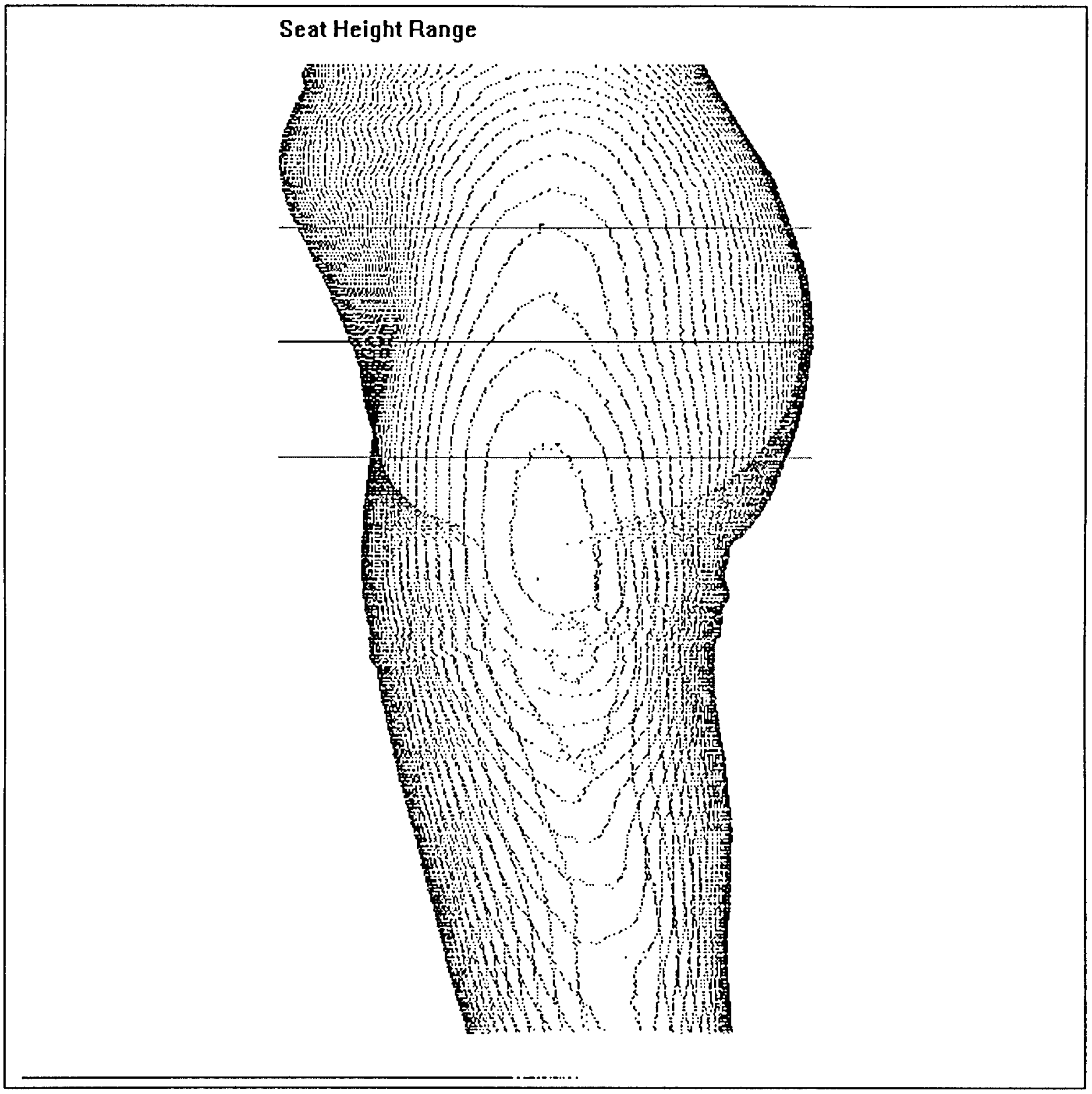
n000009.sprd.rgb.edt.ply

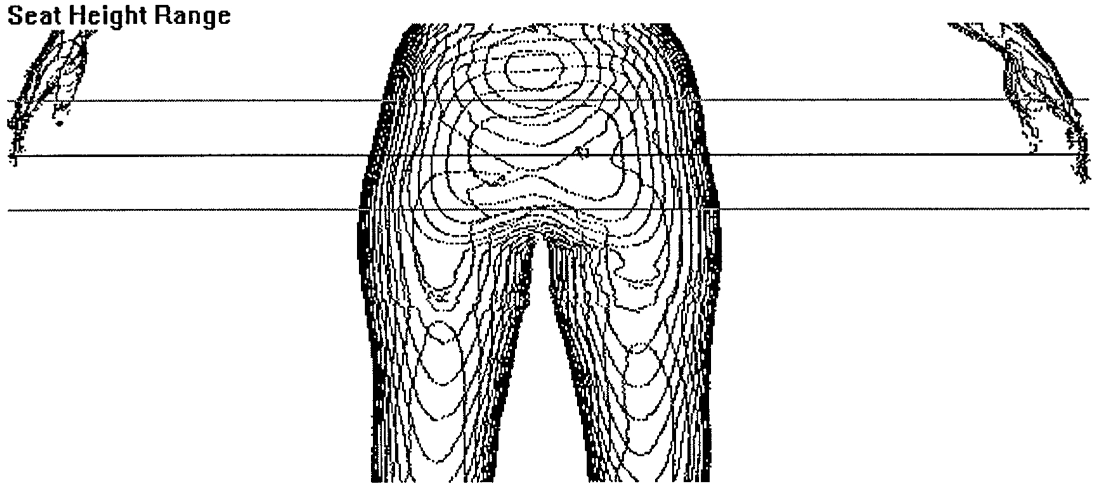
Waist Height Range

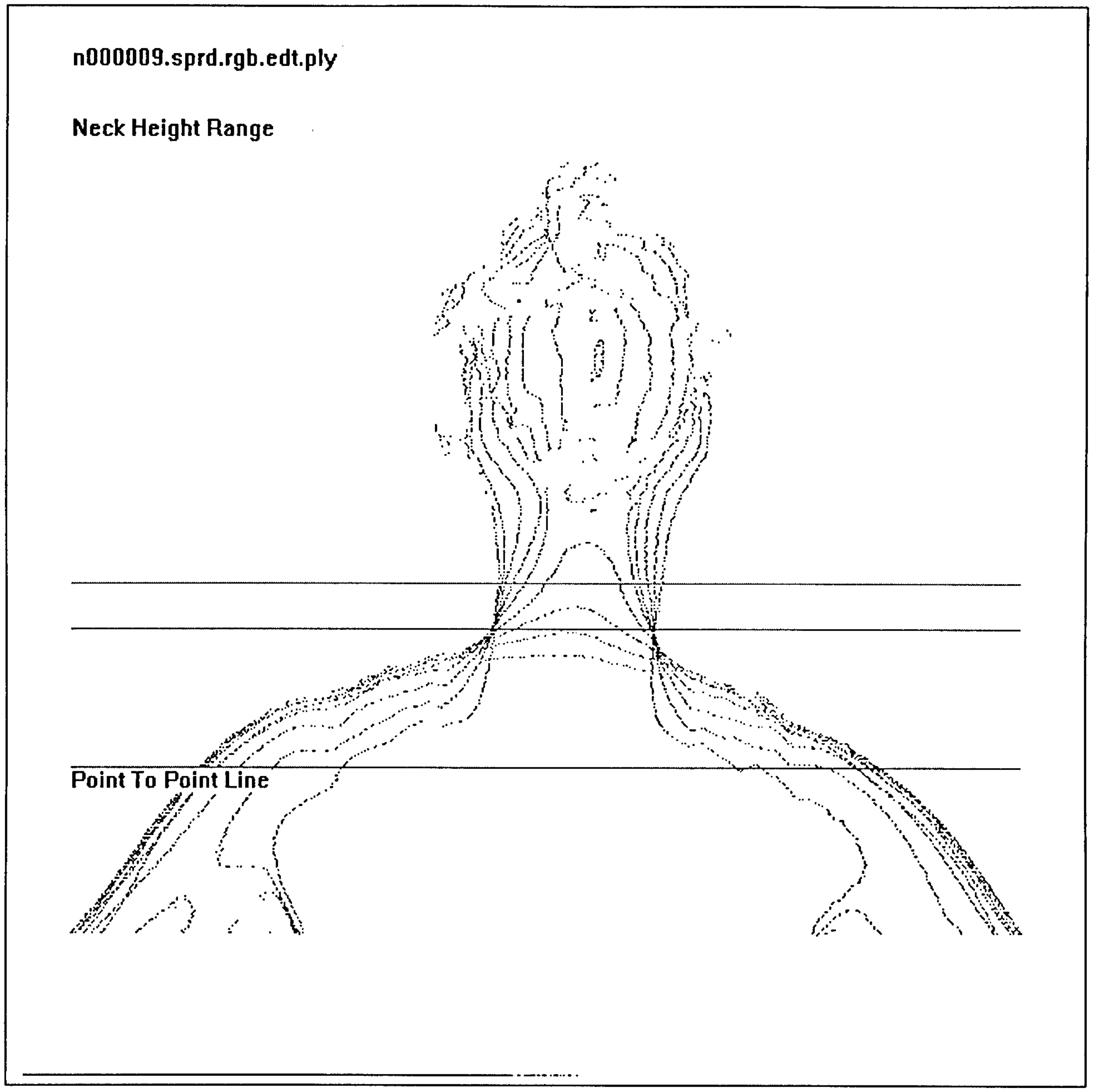
Waist = 27.837 " (707.047 ply units)

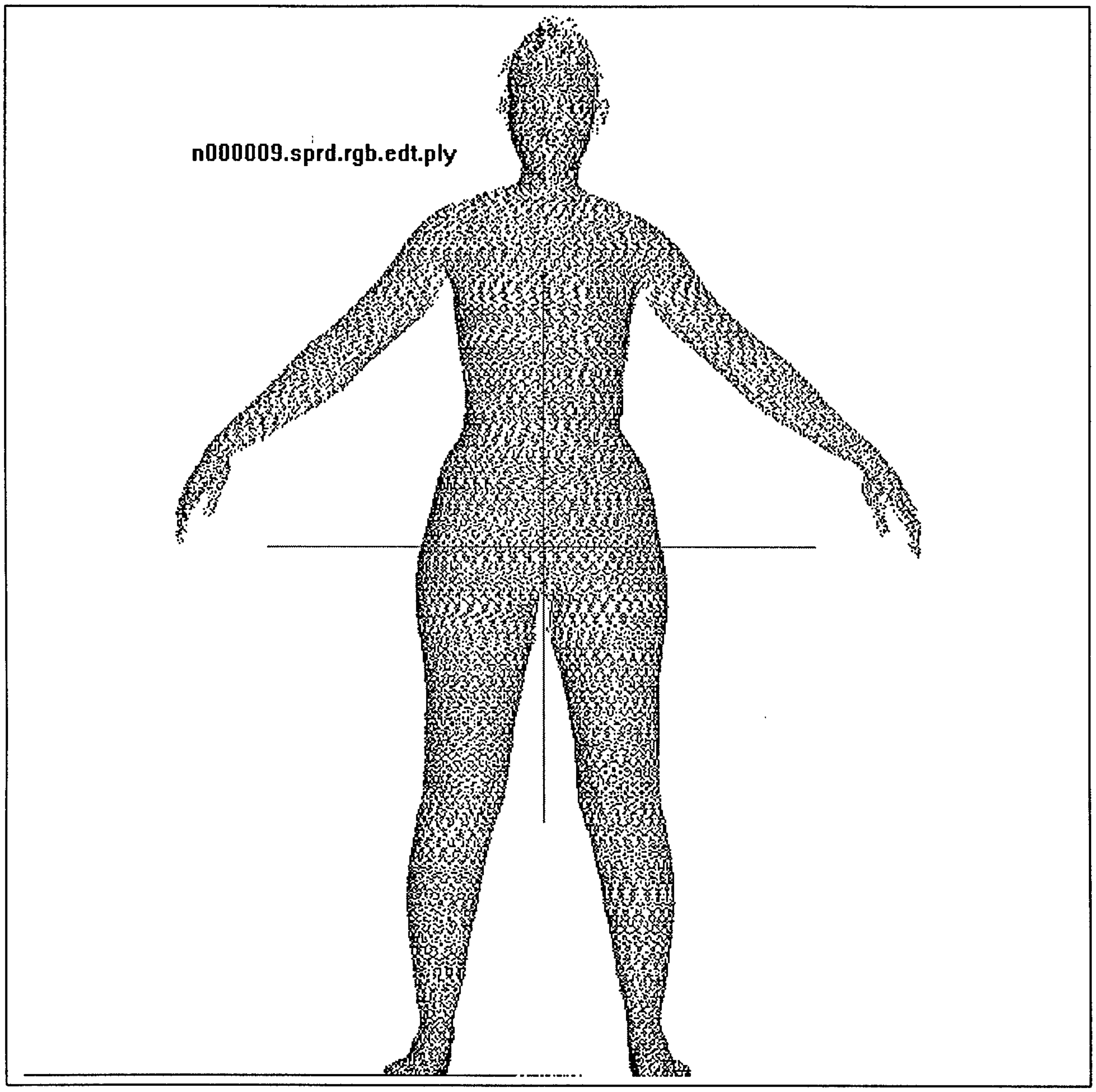


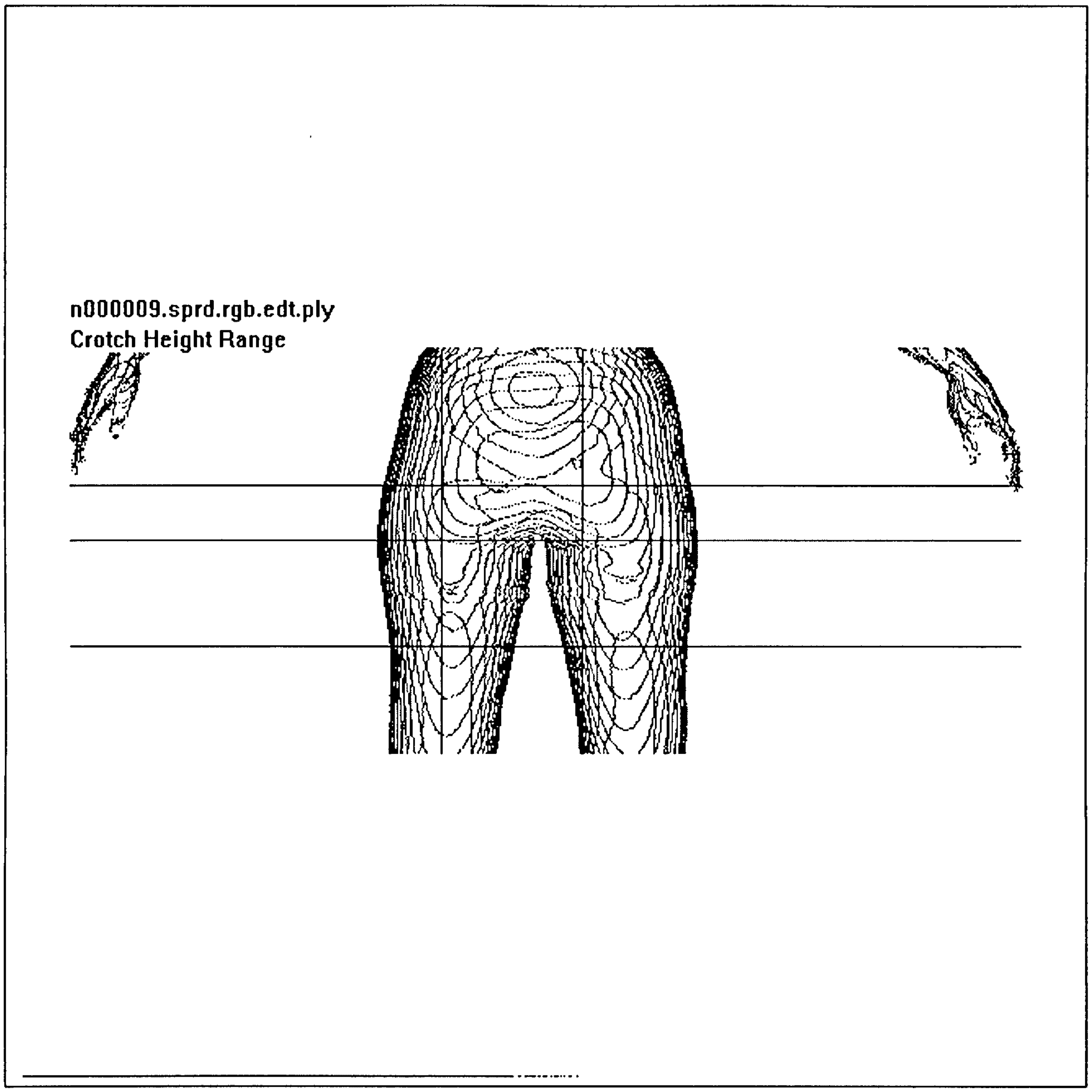

n000009.sprd.rgb.edt.ply

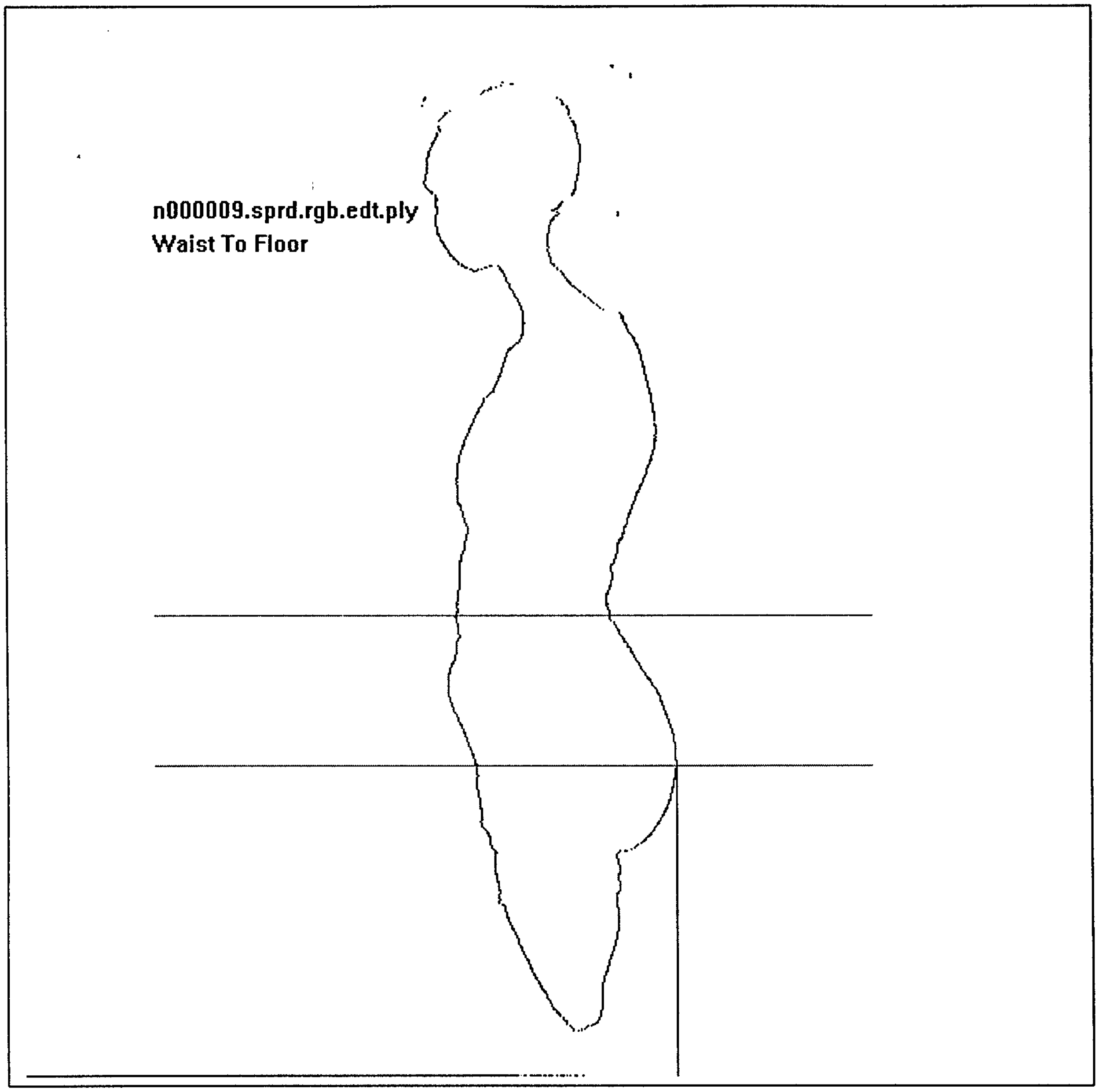

Seat Height Range


n000009.sprd.rgb.edt.ply
Seat Height Range


n000009.sprd.rgb.edt.ply
Neck Height Range
Point To Point Line


n000009.sprd.rgb.edt.ply


n000009.sprd.rgb.edt.ply
Crotch Height Range



10 Measurement Extraction Algorithms

10.1 General Human Measurements

```
float HumanMeasurements::CalcAcromialHeight()
{
#if LOGMETHODS
	cout << "HumanMeasurements::CalcAcromialHeight" << endl;
#endif
	float retVal = 0.0;
```

```
      if (pModel) {

            // At 82% of a person's height, estimate based upon Ansur data

            retVal = pModel->bounds.pmin.z + ( GetHeight() * (float)0.82 );

      }

      return retVal;
}



#if 0

float HumanMeasurements::CalcChestCircumference()
{
#if LOGMETHODS
      cout << "HumanMeasurements::CalcChestCircumference" << endl;
#endif
      float retVal = 0.0;

      if ( pModel ) {

            ams_base_type height = GetHeight();
            ams_base_type startHeight = pModel->bounds.pmin.z + (height * chestBottom);
            ams_base_type stopHeight = pModel->bounds.pmin.z + (height * chestTop);

            AMSTypes::xlistvec_type rv;
            //AMSTypes::xlistvec_type xPts( 1 );

            ams_base_type yPos;
            ams_base_type minX = pModel->bounds.pmax.x;  // largest and work backwards
            ams_base_type maxX = pModel->bounds.pmin.x;  // smallest and work forwards

            for (
                              yPos = pModel->bounds.centroid.y;
                              yPos <= pModel->bounds.centroid.y + (float)(25.4 * 10.);
                              yPos += (float)(25.4 * .25)
                        )
            {
                  pModel->findYSlice( yPos, rv );

                  int i;
                  xlist_type::const_iterator it;

                  for ( i = 0; i < rv.size(); i++ ) {
                        xlist_type & ptList = rv[i];
                        for ( it = ptList.begin() ; it != ptList.end() ; it++ ) {

                              if ( ( (*it).pt.y >= startHeight ) &&
                                          ( (*it).pt.y <= stopHeight )
                                    ) {

                                    if ( (*it).pt.x < minX ) {
                                        minX = (*it).pt.x;
                                        chestHeight = (*it).pt.y;
                                }

                                //if ( (*it).pt.x > maxX ) {
                                //      maxX = (*it).pt.x;
                                //      seatHeight = (*it).pt.y;
                                //}

                        }
                }
        }

        }

        AMSTypes::plane_type plane( AMSTypes::vector_type( 0, 0, 1 ),

    AMSTypes::point_type( 0, 0, chestHeight ) );

        // convex hulls

        pModel->doConvexHullIntersection( plane, rv );

        // make a vector with 1 xlist in it
        AMSTypes::xlistvec_type x( 1 );

        // find the hull using chest hull to eliminate the arms
        pModel->findChestHull( rv, x[0] );


        retVal = GetCircumference( x[0] );

    }

    return retVal;
}


#else // hold for new algorithm above

float HumanMeasurements::CalcChestCircumference()
{
#if LOGMETHODS
    cout << "HumanMeasurements::CalcChestCircumference" << endl;
#endif
    float retVal = 0.0;

    if (pModel) {

        ams_base_type height = GetHeight();

        // The average chest height is 72% of the overall height
        // of the subject.  Calculations based upon the 1988 ANSUR survey.

        // Based upon traditional tailoring chest height estimates, it
        // should be .71875  (i.e., 5.75 / 8 )
```

```
            float curCirc = 0.0;

            //float pcntHeight = (float)0.718;
            float pcntHeight = chestBottom;

            bool done = false;
            while ( !done ) {

                  ams_base_type ch = pModel->bounds.pmin.z + (height * pcntHeight);
                  pcntHeight += (float)0.005;

                  AMSTypes::plane_type plane( AMSTypes::vector_type( 0, 0, 1 ),

            AMSTypes::point_type( 0, 0, ch ) );

                  // convex hulls

                  AMSTypes::xlistvec_type rv;

                  pModel->doConvexHullIntersection( plane, rv );

                  // make a vector with 1 xlist in it
                  AMSTypes::xlistvec_type x( 1 );

                  // find the chest hull eliminating the arms
                  pModel->findChestHull( rv, x[0] );

                  curCirc = GetCircumference( x[0] );

                  if ( retVal <= curCirc ) {

                        retVal = curCirc;
                        chestHeight = ch;
                  }

                  if ( pcntHeight > chestTop ) {
                        done = true;
                  }
            }
      }

      return retVal;
}
#endif


float HumanMeasurements::CalcChestHeight()
{
#if LOGMETHODS
      cout << "HumanMeasurements::CalcChestHeight" << endl;
#endif
      float retVal = 0.0;

      if ( pModel ) {
            // chestHeight is set in the CalcChestCircumference routine
            // use Get  in case the value has already been set
            //
            chestCircumference = GetChestCircumference();
            retVal = chestHeight;

      }

      return retVal;
}


float HumanMeasurements::CalcCrotchHeight()
{
#if LOGMETHODS
      cout << "HumanMeasurements::CalcCrotchHeight" << endl;
#endif
      float retVal = 0.0;

      if ( pModel ) {

            // Based upon the 1988 Ansur survey, the average crotch height
            // ranges between 46.8% and 48.4% of the overall height
            // of the subject.

            // Based upon traditional tailoring crotch height estimates, it
            // should be between 45.3125%  (i.e., 3.625 / 8 ) and
            // 48.4375% (i.e., 3.875 / 8)

            ams_base_type height = GetHeight();

            float pcntHeight = crotchBottom;

            bool done = false;
            while ( !done ) {

                  ams_base_type ch = pModel->bounds.pmin.z + (height * pcntHeight);
                  pcntHeight += (float)0.0025;

                  AMSTypes::plane_type plane( AMSTypes::vector_type( 0, 0, 1 ),

            AMSTypes::point_type( 0, 0, ch ) );

                  // convex hulls

                  AMSTypes::xlistvec_type rv;

                  pModel->doConvexHullIntersection( plane, rv );


                  // throw out any "short" lists (arms/fingers/small shadows)
                  // count the major list segments remaining

                  int numMajorLists = 0;

                  for ( int i = 0 ; i < rv.size() ; i++ ) {
                    if ( xlist_ops::length( rv[ i ] ) > 152.4 ) { // 6"
                          numMajorLists++;
                    }
              }

              crotchHeight = ch;
              if ( 1 == numMajorLists ) {
                    done = true;
              }

              if ( pcntHeight > crotchTop ) {
                    done = true;
              }
        }

        retVal = crotchHeight;

  }

  return retVal;
}


float HumanMeasurements::CalcHeight()
{
#if LOGMETHODS
      cout << "HumanMeasurements::CalcHeight" << endl;
#endif
      float retVal = 0.0;

      // The highest most "z" point - the lowest "z" point
      // of the connected model.

      retVal = ( pModel->bounds.pmax.z - pModel->bounds.pmin.z );

      return retVal;
}


float HumanMeasurements::CalcNeckCircumference()
{
#if LOGMETHODS
      cout << "HumanMeasurements::CalcNeckCircumference" << endl;
#endif

      float retVal = 0.0;

      if (pModel) {

            ams_base_type height = GetHeight();

            // The average neck height is between 85.6% and 86% of the overall height
            // of the subject.  Calculations based upon the 1988 ANSUR survey.

            // Based upon traditional tailoring neck height estimates, it
            // should be between 84.375%  (i.e., 6.75 / 8 ) and 87.5% (i.e., 7 / 8 )

            // The algorithm will ranges between 85.4% and 86.2%

            float curCirc = 0.0;

            float pcntHeight = neckBottom;

            bool done = false;
            while ( !done ) {

                  ams_base_type nh = pModel->bounds.pmin.z + (height * pcntHeight);
                  pcntHeight += (float)0.002;

                  AMSTypes::plane_type plane( AMSTypes::vector_type( 0, 0, 1 ),

            AMSTypes::point_type( 0, 0, nh ) );

                  // convex hulls

                  AMSTypes::xlistvec_type rv;

                  pModel->doConvexHullIntersection( plane, rv );

                  // make a vector with 1 xlist in it
                  AMSTypes::xlistvec_type x( 1 );

                  // find the major hull!!!
                  pModel->findMajorHull( rv, x[0] );

                  curCirc = GetCircumference( x[0] );

                  if ( ( 0.0 == retVal ) || ( retVal > curCirc ) ) {

                        retVal = curCirc;
                        neckHeight = nh;
                  }

                  if ( pcntHeight > neckTop ) {
                        done = true;
                  }
            }
      }

      return retVal;
}


float HumanMeasurements::CalcNeckHeight()
{
#if LOGMETHODS
      cout << "HumanMeasurements::CalcNeckHeight" << endl;
#endif
      float retVal = 0.0;

      if ( pModel ) {
            // neckHeight is calc'd in neckCircumference
            //
            neckCircumference = GetNeckCircumference();

            retVal = neckHeight;
      }

      return retVal;
}


float HumanMeasurements::CalcOverarmCircumference()
{
#if LOGMETHODS
      cout << "HumanMeasurements::CalcOverarmCircumference" << endl;
#endif
      float retVal = 0.0;

      ams_base_type oh = GetChestHeight();

      AMSTypes::plane_type plane( AMSTypes::vector_type( 0, 0, 1 ),

      AMSTypes::point_type( 0, 0, oh ) );

      // convex hulls

      AMSTypes::xlistvec_type rv;

      pModel->doConvexHullIntersection( plane, rv );

      // make a vector with 1 xlist in it
      AMSTypes::xlistvec_type x( 1 );

      // find the major hull!!!
      pModel->findMajorHull( rv, x[0] );

      retVal = GetCircumference( x[0] );

      return retVal;
}


float HumanMeasurements::CalcPointToPoint()
{
#if LOGMETHODS
      cout << "HumanMeasurements::CalcPointToPoint" << endl;
#endif
      float retVal = 0.0;

      if (pModel) {

            // Assumes ptpHeight is one inch down from acromial height.
            // There is no equivalent measurement in the Ansur data, and
            // there is no typical tailoring height level.  Point to point
            // is determined by feel.  Roughly, it is the upper most point
```

```
		// on the outside of the shoulder.

		ams_base_type ptpHeight = GetAcromialHeight() - 25.4;

		AMSTypes::plane_type plane( AMSTypes::vector_type( 0, 0, 1 ),

	AMSTypes::point_type( 0, 0, ptpHeight ) );

		// convex hulls

		AMSTypes::xlistvec_type rv;

		pModel->doConvexHullIntersection( plane, rv );

		// make a vector with 1 xlist in it
		AMSTypes::xlistvec_type x( 1 );

		// find the major hull!!!
		pModel->findMajorHull( rv, x[0] );


		pointToPointHeight = ptpHeight;

		retVal = ( GetCircumference( x[0] ) / 2 );

	}

	return retVal;
}


float HumanMeasurements::CalcPointToPointHeight()
{
#if LOGMETHODS
	cout << "HumanMeasurements::CalcPointToPointHeight" << endl;
#endif
	float retVal = 0.0;

	if ( pModel ) {

		// pointToPointHeight is calc'd in CalcPointToPoint
		//
		pointToPoint = GetPointToPoint();

		retVal = pointToPointHeight;
	}

	return retVal;
}



float HumanMeasurements::CalcSeatCircumference()
{
#if LOGMETHODS
	cout << "HumanMeasurements::CalcSeatCircumference" << endl;
```

```
#endif
      float retVal = 0.0;

      if ( pModel ) {

            ams_base_type height = GetHeight();
            ams_base_type startHeight = pModel->bounds.pmin.z + (height * seatBottom);
            ams_base_type stopHeight = pModel->bounds.pmin.z + (height * seatTop);

            AMSTypes::xlistvec_type rv;
            //AMSTypes::xlistvec_type xPts( 1 );

            ams_base_type yPos;
            ams_base_type minX = 0.;
            ams_base_type maxX = 0.;

            for (
                              yPos = pModel->bounds.centroid.y;
                              yPos <= pModel->bounds.centroid.y + (float)(25.4 * 8.);
                              yPos += (float)(25.4 * .25)
                        )
            {
                  pModel->findYSlice( yPos, rv );

                  int i;
                  xlist_type::const_iterator it;

                  for ( i = 0; i < rv.size(); i++ ) {
                        xlist_type & ptList = rv[i];
                        for ( it = ptList.begin() ; it != ptList.end() ; it++ ) {

                              if ( ( (*it).pt.y >= startHeight ) &&
                                          ( (*it).pt.y <= stopHeight )
                                    ) {

                                    //if ( (*it).pt.x < minX ) minX = (*it).pt.x;

                                    if ( (*it).pt.x > maxX ) {

                                          maxX = (*it).pt.x;
                                          seatHeight = (*it).pt.y;

                                          maxSeatPoint.x = (*it).pt.x;
                                          maxSeatPoint.y = yPos;
                                          maxSeatPoint.z = (*it).pt.y;
                                    }
                              }
                        }
                  }
            }

            AMSTypes::plane_type plane( AMSTypes::vector_type( 0, 0, 1 ),

      AMSTypes::point_type( 0, 0, seatHeight ) );

            // convex hulls
            pModel->doConvexHullIntersection( plane, rv );

            // make a vector with 1 xlist in it
            AMSTypes::xlistvec_type x( 1 );

            if ( 1 == rv.size() ) {
                  pModel->findMajorHull( rv, x[0] );
            }
            else {
                  // find the hull using chest hull to eliminate the arms
                  pModel->findChestHull( rv, x[0] );
            }

            retVal = GetCircumference( x[0] );

      }

      return retVal;
}


#if 0 // hold for new algorithm above

float HumanMeasurements::CalcSeatCircumference()
{
#if LOGMETHODS
      cout << "HumanMeasurements::CalcSeatCircumference" << endl;
#endif
      float retVal = 0.0;

      if ( pModel ) {

            {
                  // now the cut with convex hull and majorHull routines

                  ams_base_type height = GetHeight();

                  // The average seat height is between 49% and 55% of the overall height
                  // of the subject.  Calculations based upon the 1988 ANSUR survey.

                  // For the traditional tailoring approach, the seat height is located
                  // at 53.125% (4.25 / 8) of the subject's height.

                  float curCirc = 0.0;
                  float curMaxX = 0.0;
                  float maxX = 0.0;

                  //float pcntHeight = (float)0.49;
                  float pcntHeight = seatBottom;

                  bool done = false;
                  while ( !done ) {

                        ams_base_type sh = pModel->bounds.pmin.z + (height * pcntHeight);
                        pcntHeight += (float)0.005;
                        AMSTypes::plane_type plane( AMSTypes::vector_type( 0, 0, 1 ),

                  AMSTypes::point_type( 0, 0, sh ) );

                        // convex hulls

                        AMSTypes::xlistvec_type rv;

                        pModel->doConvexHullIntersection( plane, rv );

                        // make a vector with 1 xlist in it
                        AMSTypes::xlistvec_type x( 1 );

                        // find the hull using chest hull to eliminate the arms
                        pModel->findChestHull( rv, x[0] );

#if 0
                        curCirc = GetCircumference( x[0] );

                        if (retVal == 0.0) { // first pass through loop
                              retVal = curCirc;
                        }

                        if ( retVal <= curCirc ) {

                              retVal = curCirc;
                              seatHeight = sh;
                        }
#else
                        curMaxX = GetMaxXValue( x[0] );

                        cout << "curMaxX = " << curMaxX << "  maxX = " << maxX << endl;

                        if ( (retVal == 0.0) || ( curMaxX >= maxX ) ) {
                              maxX = curMaxX;
                              curCirc = GetCircumference( x[0] );
                        }

                        cout << "retVal = " << retVal << "  curCirc = " << curCirc <<
endl;

                        if ( retVal <= curCirc ) {
                              retVal = curCirc;
                              seatHeight = sh;
                        }

#endif

                        if ( pcntHeight > seatTop ) {
                              done = true;
                        }
                  }
            }

      }
```

```
	return retVal;
}
#endif


float HumanMeasurements::CalcSeatHeight()
{
#if LOGMETHODS
	cout << "HumanMeasurements::CalcSeatHeight" << endl;
#endif
	float retVal = 0.0;

	if ( pModel ) {

		// seatHeight is calc'd in seatCircumference
		//
		seatCircumference = GetSeatCircumference();

		retVal = seatHeight;
	}

	return retVal;
}


float HumanMeasurements::CalcShoulderCircumference()
{
#if LOGMETHODS
	cout << "HumanMeasurements::CalcShoulderCircumference" << endl;
#endif
	float retVal = 0.0;

	shoulderHeight = GetPointToPointHeight() - (25.4 * .75);

	AMSTypes::plane_type plane( AMSTypes::vector_type( 0, 0, 1 ),

	AMSTypes::point_type( 0, 0, shoulderHeight ) );

	// convex hulls

	AMSTypes::xlistvec_type rv;

	pModel->doConvexHullIntersection( plane, rv );

	// make a vector with 1 xlist in it
	AMSTypes::xlistvec_type x( 1 );

	// find the major hull!!!
	pModel->findMajorHull( rv, x[0] );

	retVal = GetCircumference( x[0] );

	return retVal;
}
```

```
float HumanMeasurements::CalcShoulderHeight()
{
#if LOGMETHODS
      cout << "HumanMeasurements::CalcShoulderHeight" << endl;
#endif
      float retVal = 0.0;

      if ( pModel ) {

            // pointToPointHeight is calc'd in CalcShoulderCircumference
            //
            shoulderCircumference = GetShoulderCircumference();

            retVal = shoulderHeight;
      }

      return retVal;
}

float HumanMeasurements::CalcWaistCircumference()
{
#if LOGMETHODS
      cout << "HumanMeasurements::CalcWaistCircumference" << endl;
#endif
      float retVal = 0.0;

      if ( pModel ) {

            {
                  ams_base_type height = ( pModel->bounds.pmax.z - pModel->bounds.pmin.z
);

                  // The average waist height is between 59% and 61% of the overall
height
                  // of the subject.  Calculations based upon the 1988 ANSUR survey.

                  // Based upon traditional tailoring methods, the average waist height
is
                  // located at 59.375% (4.75 / 8) of the subject's height.

                  float curCirc = 0.0;

                  //float pcntHeight = (float)0.59;
                  float pcntHeight = waistBottom;

                  bool done = false;
                  while ( !done ) {

                        ams_base_type wh = pModel->bounds.pmin.z + (height * pcntHeight);
                        pcntHeight += (float)0.005;

                        AMSTypes::plane_type plane( AMSTypes::vector_type( 0, 0, 1 ),

                  AMSTypes::point_type( 0, 0, wh ) );
```

```
                        // convex hulls

                        AMSTypes::xlistvec_type rv;

                        pModel->doConvexHullIntersection( plane, rv );

                        // make a vector with 1 xlist in it
                        AMSTypes::xlistvec_type x( 1 );

                        // find the major hull!!!
                        pModel->findChestHull( rv, x[0] );

                        curCirc = GetCircumference( x[0] );

                        if (retVal == 0.0) retVal = curCirc;

                        if ( curCirc <= retVal ) {

                              retVal = curCirc;
                              waistHeight = wh;
                        }
                        //else {
                        //    done = true;
                        //}

                        if ( pcntHeight > waistTop ) {
                              done = true;
                        }
                  }
            }

      }

      return retVal;
}


float HumanMeasurements::CalcWaistHeight()
{
#if LOGMETHODS
      cout << "HumanMeasurements::CalcWaistHeight" << endl;
#endif
      float retVal = 0.0;

      if ( pModel ) {

            // waistHeight is set in determining waist circ.

            waistCircumference = GetWaistCircumference();
            retVal = waistHeight;
      }

      return retVal;
}
```

```
float HumanMeasurements::CalcWaistHeightBack()
{
#if LOGMETHODS
      cout << "HumanMeasurements::CalcWaistHeightBack" << endl;
#endif
      float retVal = 0.0;

      if ( pModel ) {

            // Waist height back is based upon the waist to floor
            // drop from the back of the pants.  Typically, the
            // measurement is taken in the center of the back of a
            // person wearing pants.  Because they are wearing pants
            // the cloth of the pants form a convex hull at the
            // maximum protrusion of the buttocks.  The tape measure
            // follows the curve of the buttocks to the maximum point
            // of protrusion then drops straight to the floor.

            // To match this measurement, we will need to take a slice
            // through the left or right buttock, follow the curve from
            // the waist height (retaining curve length) to its maximum
            // protrusion then calculate straight line floor drop
            // length.

#if 0
            // Currently, we are doing a straight line drop from the
            // waist height to the floor.  It does not take into
            // account the curvature of the buttock.

        retVal = GetWaistHeight() - pModel->bounds.pmin.z;
#else

            AMSTypes::xlistvec_type rv;
            AMSTypes::xlistvec_type yPts( 1 );

            ams_base_type wh = GetWaistHeight();  // waistHeight
            ams_base_type sh = GetSeatHeight();    // seatHeight

            pModel->findYSlice( maxSeatPoint.y, rv );

            int i;
            xlist_type::iterator it;

            for ( i = 0; i < rv.size(); i++ ) {
                  xlist_type & ptList = rv[i];
                  for ( it = ptList.begin() ; it != ptList.end() ; it++ ) {

                        if ( ( (*it).pt.y <= wh ) &&
                                    ( (*it).pt.y >= sh ) &&
                                ( (*it).pt.x >= pModel->bounds.centroid.x )
                              ) {

                              yPts[0].push_back( *it );

                        }
                  }
            }

            ams_base_type waistToSeat = (xlist_ops::length( yPts[0], false ) / 2);
            ams_base_type seatToFloor = AbsHeight( sh );

            retVal = (waistToSeat + seatToFloor);

#endif
      }

      return retVal;
}


float HumanMeasurements::CalcWaistHeightFront()
{
#if LOGMETHODS
      cout << "HumanMeasurements::CalcWaistHeightFront" << endl;
#endif
      float retVal = 0.0;

      if ( pModel ) {

            retVal = GetWaistHeight() - pModel->bounds.pmin.z;
      }

      return retVal;
}
```

10.2 Coat Specific Algorithms

```
float CoatMeasurements::CalcChest()
{
      float retVal = 0.0;

      retVal = hm->GetChestCircumference();

      return retVal;
}


float CoatMeasurements::CalcHeight()
{
      float retVal = 0.0;

      retVal = hm->GetHeight();

      return retVal;
}

float CoatMeasurements::CalcNeck()
{
      float retVal = 0.0;

      retVal = hm->GetNeckCircumference();

      return retVal;
}

string CoatMeasurements::CalcNeckDescription()
{
      string retVal = "";

      return retVal;
}

float CoatMeasurements::CalcOverarm()
{
      float retVal = 0.0;

      retVal = hm->GetOverarmCircumference();

      return retVal;
}

float CoatMeasurements::CalcPointToPoint()
{
      float retVal = 0.0;

      retVal = hm->GetPointToPoint();

      return retVal;
}
```

```
float CoatMeasurements::CalcPosture()
{
      float retVal = 0.0;
      return retVal;
}

string CoatMeasurements::CalcPostureDescription()
{
      string retVal = "";
      return retVal;
}

float CoatMeasurements::CalcSeat()
{
      float retVal = 0.0;

      retVal = hm->GetSeatCircumference();

      return retVal;
}

string CoatMeasurements::CalcSeatDescription()
{
      string retVal = "";
      return retVal;
}

string CoatMeasurements::CalcShoulderAngleDescription()
{
      string retVal = "";

      float delta = hm->AsInches( GetShoulderRight() );

  // ' measurement to determine the correct shoulder code
  // '
  // '  0.0   to 0.875 => extreme high  ==> shoulder code = D
  // '  1.0   to 1.375 => high          ==> shoulder code = C
  // '  1.5   to 1.625 => half high     ==> shoulder code = B
  // '  1.75  to 2.375 => regular       ==> shoulder code = A
  // '  2.5   to 2.875 => half sloping  ==> shoulder code = E
  // '  3.0   to 3.375 => full sloping  ==> shoulder code = F

      if ( delta >= 3.0 ) {
            retVal = "Full Sloping Shoulders";
            }
      else if ( delta >= 2.5 ) {
            retVal = "Half Sloping Shoulders";
            }
      else if ( delta >= 1.75 ) {
            retVal = "Regular Shoulders";
            }
      else if ( delta >= 1.5 ) {
            retVal = "Half High Shoulders";
            }
      else if ( delta >= 1.0 ) {
            retVal = "High Shoulders";
            }
      else {
            retVal = "Extreme High Shoulders";
            }
      return retVal;
}

string CoatMeasurements::CalcShoulderBuildDescription()
{
      string retVal = "";
      return retVal;
}

float CoatMeasurements::CalcShoulderLeft()
{
      float retVal = 0.0;

      float neckHeight = hm->GetNeckHeight();
      float ptpHeight = hm->GetPointToPointHeight();

      retVal = neckHeight - ptpHeight;

      return retVal;
}

string CoatMeasurements::CalcShoulderPitch()
{
      string retVal = "";
      return retVal;
}

float CoatMeasurements::CalcShoulderRight()
{
      float retVal = 0.0;

      float neckHeight = hm->GetNeckHeight();
      float ptpHeight = hm->GetPointToPointHeight();

      retVal = neckHeight - ptpHeight;

      return retVal;
}

float CoatMeasurements::CalcSleeveInseamLeft()
{
      float retVal = 0.0;
      return retVal;
}

float CoatMeasurements::CalcSleeveInseamRight()
{
      float retVal = 0.0;
      return retVal;
}

float CoatMeasurements::CalcSleeveOutseamLeft()
```

```
{
      float retVal = 0.0;
      return retVal;
}

float CoatMeasurements::CalcSleeveOutseamRight()
{
      float retVal = 0.0;
      return retVal;
}

float CoatMeasurements::CalcWaist()
{
      float retVal = 0.0;

      retVal = hm->GetWaistCircumference();

      return retVal;
}
```

10.3 Pant Specific Algorithms

```
float PantMeasurements::CalcAbdomen()
{
	float retVal = 0.0;
	return retVal;
}

float PantMeasurements::CalcCalfLeft()
{
	float retVal = 0.0;
	return retVal;
}

float PantMeasurements::CalcCalfRight()
{
	float retVal = 0.0;
	return retVal;
}

float PantMeasurements::CalcInseamLeft()
{
	float retVal = 0.0;

	retVal = mm->AbsHeight(mm->GetCrotchHeight());

	return retVal;
}

float PantMeasurements::CalcInseamRight()
{
	float retVal = 0.0;

	retVal = mm->AbsHeight(mm->GetCrotchHeight());

	return retVal;
}

float PantMeasurements::CalcKneeLeft()
{
	float retVal = 0.0;
	return retVal;
}

float PantMeasurements::CalcKneeRight()
{
	float retVal = 0.0;
	return retVal;
}

float PantMeasurements::CalcOutseamLeft()
{
	float retVal = 0.0;

	retVal = mm->AbsHeight( mm->GetWaistHeight() );
	return retVal;
}

float PantMeasurements::CalcOutseamRight()
{
	float retVal = 0.0;

	retVal = mm->AbsHeight( mm->GetWaistHeight() );

	return retVal;
}

float PantMeasurements::CalcRise()
{
	float retVal = 0.0;

	retVal = GetOutseamRight() - GetInseamRight();

	return retVal;
}

string PantMeasurements::CalcRiseDescription()
{
	string retVal = "";
	return retVal;
}

float PantMeasurements::CalcSeat()
{
	float retVal = 0.0;

	retVal = mm->GetSeatCircumference();

	return retVal;
}

string PantMeasurements::CalcSeatDescription()
{
	string retVal = "";
	return retVal;
}

float PantMeasurements::CalcThighLeft()
{
	float retVal = 0.0;
	return retVal;
}

float PantMeasurements::CalcThighRight()
{
	float retVal = 0.0;
	return retVal;
}

float PantMeasurements::CalcWaist()
```

```
{
      float retVal = 0.0;

      retVal = mm->GetWaistCircumference();

      return retVal;
}

float PantMeasurements::CalcWaistHeightBack()
{
      float retVal = 0.0;

      retVal = mm->GetWaistHeightBack();

      return retVal;
}

float PantMeasurements::CalcWaistHeightFront()
{
      float retVal = 0.0;

      retVal = mm->GetWaistHeightFront();

      return retVal;
}
```